



GEORG-AUGUST-UNIVERSITÄT  
GÖTTINGEN

## **Master's Thesis**

# **Koopman theory for data driven modeling**

prepared by

Michał Balcerak

Institute for Theoretical Physics

**First Referee: Prof. Dr. Peter Sollich (supervisor)**

**Second Referee: Prof. Dr. Steffen Schumann**

**March 2022**

# Abstract

We provide an introduction to Koopman operator theory [1] [2] [3] [4], which central problem is to find a finite-dimensional coordinate systems and embeddings in which dynamics of a given system appear approximately linear. We put it in a context of its application to time-series analysis using synthetic problems to build up intuition and real-word financial data which serves as an extremely sophisticated dynamical system with unknown evolution rule that has to be derived from data itself. Due to high complexity of the financial data, we develop appropriate feature extractions, parameter searches, and excessive anti-overfitting methods. We employ alternative time series analysis methods that are meant to serve as performance benchmarks - Gaussian Process modeling and Random Walk. Alongside regression and classification tasks, we explored interpretability of the Koopman theory derived models. We found that Koopman based DMD and benchmark methods GP, RW have its own forecasting strengths, and one can speculate that in a general case, the DMD model would be the most appropriate. Additionally, the DMD model is able to accurately classify potentially beneficial investments, even with hypothetical transaction fees. Lastly, we explored Koopman theory combined with multitask learning to discuss stationarity of the distribution of the studied data.

**Keywords:** Koopman learning, time series analysis, multitask learning, Dynamic Mode Decomposition, interpretable modeling, price forecasting

# Contents

1	Introduction . . . . .	5
2	Koopman theory . . . . .	5
2.1	An overview of Koopman theory . . . . .	6
2.2	Eigenfunctions and eigenvalues of the Koopman operator. . . . .	7
2.3	Koopman eigenfunctions and eigenvalues induced by already-linear dynamics . . . . .	7
2.4	Example . . . . .	8
3	Practical finite-dimensional representations of Koopman operator. . . . .	9
3.1	Directly from the eigenfunction equation . . . . .	9
3.2	Dynamic mode decomposition (DMD) . . . . .	10
3.3	DeepKoopman . . . . .	14
4	Other time-series analysis methods . . . . .	15
4.1	ARIMA: Auto-regressive integrating moving average . . . . .	15
4.2	Random Walk Models . . . . .	16
4.3	Gaussian Processes . . . . .	17
5	Financial data analysis . . . . .	20
5.1	Market efficiency . . . . .	20
5.2	Stationary of the market dynamics . . . . .	21

6	Experimental data and feature extraction . . . . .	21
6.1	Data sources and normalisation . . . . .	21
6.2	Price trajectories . . . . .	23
6.3	Comparison metrics . . . . .	24
6.4	Goodness of selected 2D snapshot resolution - mismatch index	25
6.5	Data pipeline . . . . .	30
7	Hyper-parameters optimisation pipeline . . . . .	30
7.1	Common parameters . . . . .	30
7.2	Model specific parameters . . . . .	31
7.3	Search algorithm . . . . .	31
8	Experimental results: sine wave with noise . . . . .	31
8.1	Data processing . . . . .	32
8.2	DMD modeling . . . . .	32
8.3	2D snapshots reconstruction . . . . .	37
9	Experimental results: financial transactions . . . . .	39
9.1	Data processing . . . . .	39
9.2	DMD modeling . . . . .	40
9.3	PDF trajectory forecasting . . . . .	42
9.4	Financial labels forecasting . . . . .	46
9.5	Volatility forecasting . . . . .	47
10	Supervised DMD experiment . . . . .	49
10.1	Seasonal labels . . . . .	49
11	Summary and conclusions . . . . .	51
11.1	Summary . . . . .	51
11.2	Conclusions . . . . .	52

# 1 Introduction

Mathematical tools and algorithms emerging from modern computing and data science transform the way dynamical systems are being studied. First-principles derivations and asymptotic reductions are giving way to data-driven approaches that formulate models in operator theoretic or probabilistic frameworks [1]. Koopman spectral theory has emerged as a dominant perspective over the past decade [2] [3] [4], in which nonlinear dynamics are represented in terms of an infinite dimensional linear operator acting on the space of all possible measurement functions of the system.

This linear representation of nonlinear dynamics has tremendous potential to enable the prediction, estimation, and control of nonlinear systems with standard textbook methods developed for linear systems. Because of simplicity of the dynamics, there is a potential for interpretation of some of dynamical system aspects, which might be critical in certain fields like finance, where the models have to be properly understood to avoid unknown risk.

However, obtaining finite-dimensional coordinate systems and embeddings in which the dynamics appear approximately linear remains a central open challenge. The success of Koopman analysis is due primarily to three key factors: 1) there exists rigorous theory 2) the approach is formulated in terms of measurements, making it ideal for leveraging big-data and machine learning techniques, and 3) simple, yet powerful numerical algorithms, such as the dynamic mode decomposition (DMD), have been developed and extended to reduce Koopman theory to practice in real-world applications [5].

In this section, we provide an introduction to Koopman operator theory based on [1] [2], and put it in a context of its application to time-series analysis using real-world financial data (also covered in [6]). The financial data serves as a complex dynamical system, and an overview of some of its challenges is given [7] [8].

Besides Koopman data-driven methods to study financial markets, we introduce alternative time series analysis methods that are meant to serve as performance benchmarks.

## 2 Koopman theory

Nonlinearity is a central challenge in dynamical systems, resulting in diverse phenomena, from bifurcations to chaos, that manifest across a range of disciplines. However, there is currently no overarching mathematical framework for the explicit and general characterisation of nonlinear systems. In contrast, linear systems are completely characterized by their spectral decomposition (i.e., eigenvalues and eigenvectors), leading to generic and computationally efficient algorithms for prediction and estimation. Importantly, linear superposition fails for nonlinear dynamical sys-

tems, leading to a variety of interesting phenomenon including frequency shifts and the generation of harmonics. The Koopman operator theory of dynamical systems provides a promising alternative perspective, where it may be possible to enable linear superposition even for strongly nonlinear dynamics via the infinite-dimensional, but linear, Koopman operator. The Koopman operator is linear, advancing measurement functions of the system, and its spectral decomposition completely characterizes the behavior of the nonlinear system. Finding tractable finite-dimensional representations of the Koopman operator is closely related to finding effective coordinate transformations in which the nonlinear dynamics appear linear.

Koopman analysis has recently gained renewed interest, in large part because of its strong connections to data-driven modeling [brunton2021modern]. In this review, we provide an overview of modern Koopman theory for dynamical systems, including an analysis of dynamic mode decomposition (DMD) method with relies on the theory and is a computational algorithm that can be applied to real-word data analysis.

### 2.1 An overview of Koopman theory

Let us consider dynamical systems of the form:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t)) \quad (1)$$

where  $\mathbf{x} \in X \subseteq \mathbb{R}^n$  is the state of the system and  $\mathbf{f}$  is a vector field describing the dynamics. In general, the dynamics may also depend on time  $t$ , although in this work we will only consider stationary dynamics or processes that can be approximated under certain conditions as stationary.

A major goal of dynamical systems analysis, is to find a transformation  $\varphi$  to a new vector of coordinates  $\tilde{\mathbf{x}} = \varphi(\mathbf{x})$  where the dynamics are simplified, or ideally, linearised:

$$\frac{d}{dt}\tilde{\mathbf{x}} = \mathbf{L}\tilde{\mathbf{x}} \quad (2)$$

What Koopman theory based methods are trying to achieve is to unfold nonlinearities of the dynamics by allowing higher-dimensional data representations.

In practice, we typically have access to discrete measurements, rather than continuous states. This type of data is governed by the discrete-time dynamical system (*flow map*):

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k) \quad (3)$$

where  $\mathbf{x}_k = \mathbf{x}(t_k) = \mathbf{x}(k\Delta t)$ .

Now the goal is to find a coordinate transformation such that the discrete dynamical system can be approximated by a linear system:

ics in the new representation become linear. These coordinates are eigenfunctions of the Koopman operator  $\mathcal{K}$ , which by definition advances measurement functions  $g(\mathbf{x})$  forward in time in the following way:

$$\mathcal{K}g(\mathbf{x}_k) := g(\mathbf{F}(\mathbf{x}_k)) \quad (4)$$

This has to be true for *any* measurement function  $g$  and for any state  $\mathbf{x}_k$ .

We observe that  $\mathcal{K}$  acts linearly on vector of measurements of the state  $\mathbf{g}$ , even though the dynamics defined by  $\mathbf{F}$  may be nonlinear.

For an eigenfunction  $\varphi$  of  $\mathcal{K}$ , and its corresponding eigenvalue  $\lambda$ , the advancement in time becomes:

$$\mathcal{K}\varphi(\mathbf{x}_k) = \lambda\varphi(\mathbf{x}_k) = \varphi(\mathbf{x}_{k+1}) \quad (5)$$

Measurements that can be formed as linear combinations of eigenfunctions have a particular simple evolution in time:

$$g(\mathbf{x}) = \sum_i a_i \varphi_i \Rightarrow \mathcal{K}^t g(\mathbf{x}) = \sum_i a_i \lambda_i^t \varphi_i \quad (6)$$

## 2.2 Eigenfunctions and eigenvalues of the Koopman operator.

A Koopman eigenfunction  $\varphi(\mathbf{x})$  corresponding to an eigenvalue  $\lambda$  fulfills:

$$\varphi(\mathbf{x}_{k+1}) = \mathcal{K}\varphi(\mathbf{x}_k) = \lambda\varphi(\mathbf{x}_k) \quad (7)$$

In continuous time:

$$\frac{d}{dt}\varphi(\mathbf{x}) = \mathcal{L}\varphi(\mathbf{x}) = \mu\varphi(\mathbf{x}) \quad (8)$$

where  $\mu = \log \lambda$ .

## 2.3 Koopman eigenfunctions and eigenvalues induced by already-linear dynamics

Let us calculate eigenfunctions of  $\mathcal{K}, \mathcal{L}$  that are induced by already-linear discrete dynamics  $\mathbf{x}_{n+1} = \mathbf{A}\mathbf{x}_n$ .

Given a *left*-eigenvector:  $\boldsymbol{\Omega}^T \mathbf{A} = \lambda \boldsymbol{\Omega}^T$ , we can construct a Koopman eigenfunction as follows:

$$\varphi(\mathbf{x}) := \boldsymbol{\Omega}^T \mathbf{x} \quad (9)$$

because now:

$$\mathcal{K}\varphi(\mathbf{x}) = \varphi(\mathbf{Ax}) = \boldsymbol{\Omega}^T \mathbf{Ax} = \lambda \boldsymbol{\Omega}^T \mathbf{x} = \lambda \varphi(\mathbf{x}) \quad (10)$$

## 2.4 Example

After mathematically expressing the objective of Koopman theory, let us continue with an example to illustrate the linearisation of dynamics in higher-dimensional representation in practice.

The example dynamical system[5] is given by:

$$\dot{x}_1 = \mu x_1 \quad (11)$$

$$\dot{x}_2 = \lambda(x_2 - x_1^2) \quad (12)$$

It is possible to augment the state  $\mathbf{x}$  with the nonlinear measurement  $x_3 = x_1^2$ . In these coordinates, the dynamics become linear:

$$\frac{d}{dt} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \mu & 0 & 0 \\ 0 & \lambda & -\lambda \\ 0 & 0 & 2\mu \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (13)$$

We can extract the Koopman eigenfunction directly from the matrix in (13). The *left*-eigenvectors of the matrix corresponding to  $\mu, 2\mu, \lambda$ :

$$\Omega_\mu^T = [1 \quad 0 \quad 0] \quad (14)$$

$$\Omega_\lambda^T = [0 \quad 1 \quad -b] \quad \text{with} \quad b = \frac{\lambda}{\lambda - 2\mu} \quad (15)$$

$$\Omega_{2\mu}^T = [0 \quad 0 \quad 1] \quad (16)$$

We can now calculate the correspond eigenfunctions of the Koopman operator using (9):

$$\varphi_\mu = x_1 \quad (17)$$

$$\varphi_\lambda = x_2 - bx_1^2 \quad \text{with} \quad b = \frac{\lambda}{\lambda - 2\mu} \quad (18)$$

$$\varphi_{2\mu} = x_1^2 \quad (19)$$

$$(20)$$



Relation between  $\varphi_\mu$  and  $\varphi_{2\mu}$  is an illustration of the fact that in general, product of two eigenfunctions  $\varphi_{\lambda_1}$  and  $\varphi_{\lambda_2}$  is also an eigenfunction to eigenvalue  $\lambda_1\lambda_2$ .

Using a linear combination of the Koopman operator eigenfunctions we could now construct measurements of the system that would remain invariant under the Koopman operator (time translation).

### 3 Practical finite-dimensional representations of Koopman operator.

We now see that the dynamics in eigenfunction coordinates space will be linear, and represented by a matrix if we are able to find finite-dimensional approximations of the infinite-dimensional Koopman operator with eigenfunctions that span our desired measurements. This has proven challenging in practical applications. In this chapter, we discuss some of the approaches some of the approaches to the eigenfunction extraction problem.

#### 3.1 Directly from the eigenfunction equation

Many approaches seek to identify eigenfunctions of the Koopman operator directly from (5). We can rewrite (5) using Koopman operator generator for continuous systems,  $\mathcal{L}$ :

$$\frac{d}{dt}g(\mathbf{x}) = \mathcal{L}g(\mathbf{x}) \quad (21)$$

One can acquire the PDE for Koopman eigenfunctions this way:

$$\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x}) \quad \frac{d}{dt}\varphi(\mathbf{x}) = \mu\varphi(\mathbf{x}) \quad (22)$$

$$\frac{d}{dt}\varphi(\mathbf{x}) = \mathbf{f}(\mathbf{x})\nabla\varphi(\mathbf{x}) \quad (23)$$

Which yields the PDE:

$$\mathbf{f}(\mathbf{x})\nabla\varphi(\mathbf{x}) = \mu\varphi(\mathbf{x}) \quad (24)$$

With this PDE, it is possible to approximate eigenfunctions, either by solving for the Laurent series or with data via regression. However, the resulting models can be of exceedingly high dimension.

### 3.2 Dynamic mode decomposition (DMD)

Dynamic mode decomposition [4] is a method to acquire finite-dimensional representations of the Koopman operator by identifying spatio-temporal coherent structures from a high-dimensional dynamical system. In this method, we restrict ourselves only to linear measurements of the system,  $g(\mathbf{x}) = \mathbf{x}$ , which does not capture nonlinear effects of dynamics by itself. We will combine this method with a state dimensionality expansion to capture some of the nonlinearities of the dynamics of the original states; the same way non-markovian systems can be transformed to markovian.

DMD is based on a computationally efficient singular value decomposition (SVD), so that it provides scalable dimensionality reduction for high-dimensional data. The SVD orders modes hierarchically based on how much of variance of the original data is captured by each mode. Because of the variance hierarchy, dimensionality reduction aspect might be used not only to make the algorithms computationally efficient, but also to denoise, and prevent overfitting. The DMD modes are linear combinations of the SVD modes that are chosen specifically to extract spatially correlated structures that have the same coherent linear behaviour in time, given by oscillations at a fixed frequency with growth or decay. The additional interpretability aspect of the dynamics can be much appreciated in many fields, for example finance.

The DMD algorithm seeks a best-fit linear operator  $A$  that approximately advances the state of a system  $x \in R^n$ , forward in time according to the linear dynamical system:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k \quad (25)$$

where  $\mathbf{x}_k = \mathbf{x}(t_k) = \mathbf{x}(k\Delta t)$ , and  $\Delta t$  is a fixed time step small enough to capture highest frequencies in the dynamics we wish to model. Thus, the operator  $A$  is an approximation of the Koopman operator  $K$  restricted to direct measurements of the state  $\mathbf{x}$ .

DMD is fundamentally a data-driven algorithm, and the operator  $\mathbf{A}$  is approximated from a collection of snapshots:  $(\mathbf{x}_1, \mathbf{x}_1, \dots, \mathbf{x}_m)$ . A snapshot is typically a measurement of the full state of the system, such as a fluid velocity field sampled at a large number of spatially discretised locations, that is reshaped into a column vector of high dimension. In case of time-series analysis, it can be a vector representing a trajectory in a similar fashion. We use the original Schmid's formulation [4] that requires uniform sampling in time. However, the *exact* DMD algorithm of Tu et al. [3], may be applied to irregularly spaced data and data concatenated from multiple different time series. The snapshots are arranged into two data matrices,  $\mathbf{X}$  and  $\mathbf{X}'$ :

$$\mathbf{X} = \begin{bmatrix} | & | & \dots & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{m-1} \\ | & | & \dots & | \end{bmatrix} \quad (26)$$

$$\mathbf{X}' = \begin{bmatrix} | & | & \dots & | \\ \mathbf{x}_2 & \mathbf{x}_3 & \dots & \mathbf{x}_m \\ | & | & \dots & | \end{bmatrix}. \quad (27)$$

Equation (25) can be written in terms of these matrices as:

$$\mathbf{X}' \approx \mathbf{A}\mathbf{X}. \quad (28)$$

The best fit operator  $A$  establishes a linear dynamical system that approximately advances snapshots forward in time. It can be formulated as a least squares optimisation problem in the following way:

$$\mathbf{A} = \underset{\mathbf{A}}{\operatorname{argmin}} \|\mathbf{X}' - \mathbf{A}\mathbf{X}\|_F^2 = \mathbf{X}'\mathbf{X}^\dagger \quad (29)$$

where  $\|\cdot\|_F$  is the Frobenius norm and  $X^\dagger$  is the pseudo-inverse of  $X$ . The DMD algorithm is illustrated on Figure 1.

### SVD-based algorithm for rank- $r$ approximation

The SVD-based algorithm presented in [4] is generally accepted as the defining DMD algorithm. In general, any  $m \times n$  matrix can be decomposed as  $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$ , where  $*$  is the complex-conjugate transpose, matrices  $\mathbf{U} \in \mathbb{C}^{n \times n}$ ,  $\mathbf{V} \in \mathbb{C}^{m \times m}$  are unitary, and  $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$  is a rectangular diagonal matrix with non-negative real numbers on the diagonal. The diagonal entries  $\sigma_i = \Sigma_{ii}$  of  $\mathbf{\Sigma}$  are uniquely determined by  $\mathbf{X}$  and are known as the singular values of  $\mathbf{X}$ . The number of non-zero singular values is equal to the rank of  $\mathbf{X}$ . The columns  $\mathbf{u}_i$  and  $\mathbf{v}_i$  are called left-singular and right-singular vectors of  $\mathbf{X}$ , respectively. They form two sets of orthonormal bases and the singular value decomposition can be written as  $\mathbf{X} = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^*$  where  $r \leq \min(m, n)$  is the rank of  $\mathbf{X}$ .

The pseudo-inverse from (29) can be computed using the SVD of  $\mathbf{X}$  as  $\mathbf{X}^\dagger = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^*$ . The SVD-based algorithm seeks only the leading eigenvectors and eigenvalues of  $\mathbf{A}$  without explicitly constructing it. This makes the algorithm more computationally efficient. Instead of computing  $\mathbf{A}$  in (29), we can first project  $\mathbf{A}$  onto the leading  $r$  SVD modes (columns of  $\mathbf{U}$ )  $\mathbf{U}_r$ , and compute the pseudo-inverse using the rank- $r$  SVD approximation  $\mathbf{X} \approx \mathbf{U}_r \mathbf{\Sigma}_r \mathbf{V}_r^*$ :

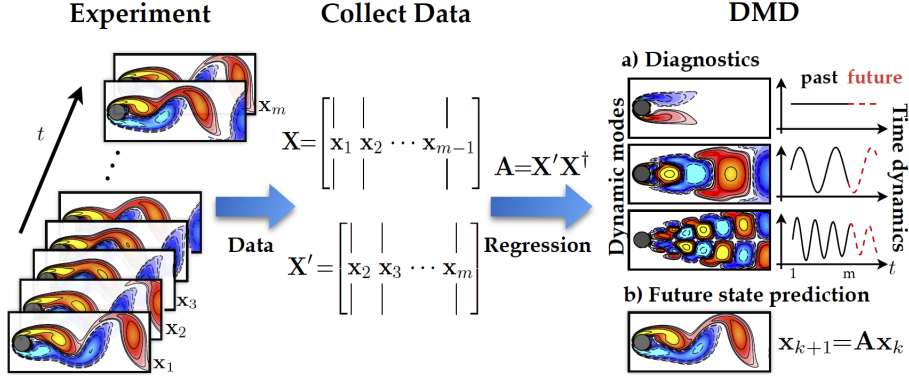


Figure 1: Overview of the DMD algorithm illustrated on the fluid flow past a circular cylinder. *Reproduced from Kutz et al. [1].*

$$\tilde{\mathbf{A}} = \mathbf{U}_r^* \mathbf{A} \mathbf{U}_r = \mathbf{U}_r^* \mathbf{X}' \mathbf{X}^\dagger \mathbf{U}_r = \mathbf{U}_r^* \mathbf{X}' \mathbf{V}_r \Sigma_r^{-1} \mathbf{U}_r^* \mathbf{U}_r = \mathbf{U}_r^* \mathbf{X}' \mathbf{V}_r \Sigma_r \quad (30)$$

The leading eigenvectors and eigenvalues of  $\mathbf{A}$  can be now computed using the spectral decomposition of a relatively small matrix  $\tilde{\mathbf{A}}$ :

$$\tilde{\mathbf{A}} \mathbf{W} = \mathbf{W} \mathbf{\Lambda} \quad (31)$$

The diagonal matrix  $\mathbf{\Lambda}$  contains *DMD eigenvalues* corresponding to eigenvalues of higher dimensional  $\mathbf{A}$ .

The eigenvectors of  $A$  are the *DMD modes*  $\Phi$ , and can be constructed using the eigenvectors  $\mathbf{W}$  of the projected matrix  $\tilde{\mathbf{A}}$  and the time-shifted data matrix  $\mathbf{X}'$  (proved in [2] under certain conditions):

$$\Phi = \mathbf{X}' \mathbf{V}_r \Sigma_r^{-1} \mathbf{W} \quad (32)$$

The algorithm, resulting DMD modes, and their dynamics induces by DMD eigenvalues are illustrated on Figure 1.

### DMD state representation

Once we have DMD modes and eigenvalues, we can use them to represent the system state:

$$\mathbf{x}_k = \sum_{i=1}^r \phi_i \lambda_i^{k-1} b_i = \Phi \mathbf{\Lambda}^{k-1} \mathbf{b} \quad (33)$$

where  $\phi_i$  are eigenvectors of  $\mathbf{A}$  (DMD modes),  $\lambda_i$  are eigenvalues of  $\mathbf{A}$  (DMD eigenvalues), and  $b_i$  are the mode amplitudes induced by initial conditions. They can be given by:

$$\mathbf{b} = \Phi^\dagger \mathbf{x}_1 \quad (34)$$

### Sparsity promoting DMD

Alternatively to Eq. 34, using the DMD state representation 33, we can acquire the mode amplitudes as a solution to another optimisation problem:

$$\underset{\mathbf{b}}{\operatorname{argmin}} \|\mathbf{X} - \Phi \mathbf{D}_{\mathbf{b}} \mathbf{M}\|_F^2 \quad (35)$$

where  $\mathbf{D}_{\mathbf{b}} := \operatorname{diag}(b_i)$ , and  $\mathbf{M}$  stands for a Vandermonde matrix of the series of eigenvalues:

$$\mathbf{M} = \begin{bmatrix} 1 & \lambda_1 & \dots & \lambda_1^{m-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_r & \dots & \lambda_r^{m-1} \end{bmatrix} \quad (36)$$

Jovanović et al. [9] developed the first algorithm to improve the estimate of the modal amplitudes by promoting sparsity. In this case, the underlying optimisation algorithm is centered around improving the approximation  $\mathbf{X} \approx \Phi \mathbf{D}_{\mathbf{b}} \mathbf{M}$  using the formulation:

$$\underset{\mathbf{b}}{\operatorname{argmin}} \|\mathbf{X} - \Phi \mathbf{D}_{\mathbf{b}} \mathbf{M}\|_F^2 + \gamma \|\mathbf{b}\|_{L_1} \quad (37)$$

where  $\gamma$  a regularisation parameter, and  $\|\cdot\|_{L_1}$  is the  $L_1$ -norm. The globally optimal solution of the resulting regularized convex optimisation problem can be computed using the alternating direction method of multipliers [9].

### Supervised DMD via multitask learning

DMD is in principle an unsupervised algorithm, and is not equipped with a mechanism to utilize label information even if a set of data with different labels is given. Keisuke Fuji and Yoshinobu Kawahara introduced a supervised version of the DMD algorithm via multitask learning [10][11].

Multitask learning is an approach that aims to improve generalisation by using domain information contained in training signals of many tasks. It does this by learning tasks in parallel while using a shared representation across them; what is learned for each task can help other tasks be learned better. If we represent an optimisation problems as tasks, instead of solving them individually, we combine them to form a single optimisation problem, in a way that the difficulty can be shared across different tasks. This requires usage of norms that cannot be separated back into a sum of metrics of individual tasks, and enforces some kind of shared representation across the optimisation problems.

Suppose that we have  $N$  sequences of length  $\tau$  with labels  $\{1, \dots, K\}$ , and let:  $X_{n,0} = [x_{n,1}, x_{n,2}, \dots, x_{n,\tau-1}]$  and  $X_{n,1} = [x_{n,2}, x_{n,3}, \dots, x_{n,\tau}]$ . We assume that there

are the common or label-specific (or mixed) dynamical structures (i.e., frequencies with decay/growth rate) behind the sequences.

First, DMD is performed for all labels using vertically arranged data matrices at every sequence (defined as  $\mathbf{X}_0$  and  $\mathbf{X}_1$  for arranged  $\mathbf{X}_{n,0}$  and  $\mathbf{X}_{n,1}$  respectively) to obtain initial DMD solution  $\mathbf{A}$  and eigenvalue matrix  $\mathbf{\Lambda}$  (and also  $\mathbf{U}$ ,  $\mathbf{V}$ , and  $\mathbf{W}$  just like in the SVD-based algorithm), which are common among all labels. We then introduce sparse regularisation for each label with the common DMD eigenvalue matrix  $\mathbf{\Lambda}$  and thereafter update the DMD eigenvector matrix  $\mathbf{W}$  to be label specific, thus acquiring  $\mathbf{W}_k$  for each label  $k$  (details of this procedure are in the original paper [11]). Then, we consider DMD using concatenated data matrices  $\mathbf{X}_0^{(k)}$  and  $\mathbf{X}_1^{(k)}$  vertically arranging data matrices  $\mathbf{X}_{n,0}$  and  $\mathbf{X}_{n,1}$  with label  $k$ , respectively. We can now formulate a similar optimisation problem to the equation 40, but with embedded label specific sparse regularisation:

$$\operatorname{argmin} \left\| \mathbf{U}_k^* \mathbf{X}_1^k - \mathbf{W}_k \mathbf{D}_{\mathbf{b}_k} \mathbf{M}_1 \right\|_F^2 \quad (38)$$

where  $\mathbf{D}_{\mathbf{b}_k}$  is a diagonal matrix arranging sparse representation  $\mathbf{b}$  with label  $k$ ,  $\mathbf{W}_k$  is the DMD eigenvector matrix for label  $k$ , and  $\mathbf{M}_1$  is the time-shifted Vandermonde matrix:

$$\mathbf{M}_1 = \begin{bmatrix} \lambda_{n,1} & \lambda_{n,1}^2 & \cdots & \lambda_{n,1}^\tau \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{n,r} & \lambda_{n,r}^2 & \cdots & \lambda_{n,r}^\tau \end{bmatrix} \quad (39)$$

In addition, the method introduces sparse-group Lasso regularisation [**sparsegroup**], which seeks both sparsity of groups and within each group. In this case, we obtain group sparsity for each eigenvalue dimension  $d = \{1, \dots, r\}$  among all labels to extract common dynamical structures. Therefore, the objective function becomes:

$$\operatorname{argmin}_{b_{k,j}, w_{k,j}} \left\| \mathbf{U}_k^* \mathbf{X}_1^k - \mathbf{W}_k \mathbf{D}_{\mathbf{b}_k} \mathbf{M}_1 \right\|_F^2 + \gamma_1 \sum_{k=1}^K \sum_{j=1}^r |b_{k,j}| + \gamma_2 \sum_{j=1}^r \|\mathbf{b}_j\| \quad (40)$$

where  $w_{k,j}$  and  $b_{k,j}$  are components of  $\mathbf{W}_k$  and  $\mathbf{b}$  for label  $k$  and dimension  $j$ , respectively.  $\mathbf{b}_j$  is a vector arranging  $b_{k,j}$  among all labels and  $\|\cdot\|_2$  is the  $L_2$ -norm of a given vector. The  $\gamma_1$  and  $\gamma_2$  are hyper-parameters. This optimisation problem is called supervised DMD.

### 3.3 DeepKoopman

The challenge of identifying and representing Koopman eigenfunctions also provides motivation for the use of emerging deep learning methods. [12],[13], [14] In all of

these recent studies, DNN representations have been shown to be flexible, exhibit high accuracy, and challenge other leading methods.

### Network layout

The goal of DeepKoopman [15], method is to utilise DNN representations of Koopman eigenfunctions that match the intrinsic low-rank dynamics while actively trying to avoid overfitting and remaining interpretable, thus merging the best of DNN architectures and Koopman theory. The network layout is illustrated on Figure 2.

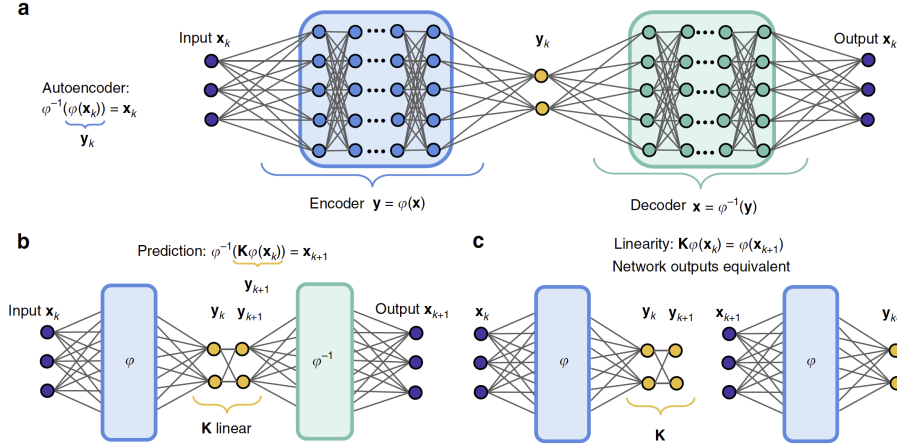


Figure 2: Diagram of the deep learning schema to identify Koopman eigenfunctions  $\varphi(x)$ . **a** The network is based on a deep auto-encoder, which is able to identify intrinsic coordinates  $y = \varphi(x)$  and decode these coordinates to recover  $x = \varphi^{-1}(y)$ . **b, c** We add an additional loss function to identify a linear Koopman model  $K$  that advances the intrinsic variables  $y$  forward in time. In **b**, the loss function is evaluated on the state variable  $x$  and in **c** it is evaluated on  $y$ . The Figure taken directly from [15].

## 4 Other time-series analysis methods

In this section, we cover some of the methods of time-series forecasting that do not involve Koopman spectral analysis, which can be used as a benchmark.

### 4.1 ARIMA: Auto-regressive integrating moving average

Autoregressive moving average (ARMA) models are linear models which generalised autoregressive integrated moving average (ARIMA) models [16] and are commonly used in statistics and econometrics. This acronym is descriptive, capturing the key aspects of the model itself. They are:

- **AR** Autoregression. A model that uses the dependent relationship between an observation and some number of lagged observations.
- **I** Integrated. The use of differencing of raw observations in order to make the time series stationary.
- **MA** Moving Average. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

These models leverage time-series data to build models to forecast predictions into the future. ARMA and ARIMA models are characterized by a number of key parameters, one of them being the number of past time points used for forecasting a future point. However, DMD correlates each time snapshot directly to the previous time snapshot.

## 4.2 Random Walk Models

Consider a time-series  $S_n = S(n\Delta t)$ . For some financial time-series, we expect the value to grow exponentially, so that  $\log S_n$  grows linearly in time. In Random Walk Models, we also require a random component in the evolution.

The resulting model for  $S_n$  is:

$$\log S_{n+1} = \log S_n + \alpha + \beta d_{n+1} \quad (41)$$

in which  $\alpha$  and  $\beta$  are constants and  $d_n$  is a random variable. Through recursion, we can obtain:

$$\log S_{n+1} = \log S_n + n\alpha + \beta \sum_{i=1}^n d_i \quad (42)$$

which can be exponentiated to get:

$$S_n = S_0 \exp(n\alpha + \beta \sum_{i=1}^n d_i) = S_0 \exp(\gamma t_n + \beta \sum_{i=1}^n d_i) \quad (43)$$

where  $\alpha = \gamma\Delta t$ . In financial literature [**brownian**], the random variables  $d_n$  are IID  $N(0, 1)$ . Then:

$$S_n = S_0 \exp(\gamma t_n + \beta x_n) \quad (44)$$

in which  $x_n$  is a random walk with Gaussian increments, meaning  $x_n = \sqrt{n}\omega_n$  where  $\omega_n$  is  $N(0, 1)$ . Now let us set  $\mu = \gamma + \sigma^2/2$  and  $\beta = \sigma\sqrt{\Delta t}$ . Given the substitutions, let us now calculate the expected value of  $S_n$ :

$$E[S_n] = S_0 \exp((\mu - \sigma^2/2)t_n) E[\exp(\sigma\sqrt{t_n}\omega_n)] \quad (45)$$

$$= S_0 \exp((\mu - \sigma^2/2)t_n) \exp((\sigma^2/2)t_n) \quad (46)$$

$$= S_0 \exp(\mu t_n) \quad (47)$$



Equation 47 gives interpretation to  $\mu$ , which can be now seen as a growth rate coefficient. The constant  $\sigma$ , which is called the volatility, measures the amount of deviation from the expected trajectory (in finance seen as *risk*). In particular:

$$\text{var}(\log S_n) = E[(\log S_n - E[\log S_n])^2] \quad (48)$$

$$= \sigma^2 t_n E[\omega_n^2] \quad (49)$$

$$= \sigma^2 t_n \quad (50)$$

This shows that the variance of  $\log S_n$  grows linearly in time with a coefficient  $\sigma^2$ .

### 4.3 Gaussian Processes

Gaussian process models utilise probabilistic inference, which takes a group of hypotheses (models) and weights those hypotheses based on how well their predictions match the data. This approach is appealing for two reasons. First, keeping all hypotheses that match the data helps to guard against over-fitting. Second, comparing how well a dataset is fit by different models gives a way of finding which sorts of structure are present in that data.

#### Definition

A Gaussian process (GP) is any distribution over functions such that any finite set of function values  $f(\mathbf{x}_1), f(\mathbf{x}_2), \dots, f(\mathbf{x}_N)$  have a joint Gaussian distribution [17]. A GP model, before conditioning on data, is completely specified by its mean function,

$$\mathbb{E}[f(\mathbf{x})] = \mu(\mathbf{x}) \quad (51)$$

and its covariance function called the *kernel*:

$$\text{Cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}') \quad (52)$$

In practice, we can assume that the mean function is simply zero everywhere, because uncertainty about the mean function can be taken into account by adding an extra term to the kernel. Since we took out the mean, the kind of structure that can be captured by a GP model is entirely determined by its kernel. The kernel determines how the model generalizes, or extrapolates to unseen data.

One of the main difficulties in using Gaussian process modelling is constructing a kernel which represents the particular structure that is present in the data being modelled.

### Model selection

The crucial property of GPs that allows us to automatically construct models is that we can compute the *marginal likelihood* of a dataset given a particular model. The *marginal likelihood* allows one to compare models, balancing between the complexity of a model and its fit to the data. The marginal likelihood under a GP prior of a set of function values  $\mathbf{f}(\mathbf{X}) := [f(\mathbf{x}_1), f(\mathbf{x}_2), \dots, f(\mathbf{x}_N)]$  at locations  $\mathbf{X}$  is given by:

$$p(\mathbf{f}(\mathbf{X})|\mathbf{X}, \mu(\cdot), k(\cdot, \cdot)) = \mathcal{N}(\mathbf{f}(\mathbf{X})|\mathbf{X}, \mu(\mathbf{X}), k(\mathbf{X}, \mathbf{X})) \quad (53)$$

This multivariate Gaussian density is referred to as the marginal likelihood because it implicitly integrates (marginalizes) over all possible functions values  $f(\tilde{X})$ , where  $\tilde{X}$  is the set of all locations where we have not observed the function.

### Prediction

We can ask the model which function values are likely to occur at any location, given the observations seen so far. By the formula for Gaussian conditionals [17] the predictive distribution of a function value  $f(x^*)$  at a test point  $x^*$  has the form:

$$\begin{aligned} p(f(\mathbf{x}^*)|\mathbf{X}, \mu(\cdot), k(\cdot, \cdot)) &= \mathcal{N}(f(\mathbf{x}^*)|\mathbf{X}, \\ &\mu(\mathbf{x}^*) + k(\mathbf{x}^*, \mathbf{X})k(\mathbf{X}, \mathbf{X})^{-1}(\mathbf{f}(\mathbf{X}) - \mu(\mathbf{X})), \\ &k(\mathbf{x}^*, \mathbf{x}^*) - k(\mathbf{x}^*, \mathbf{X})k(\mathbf{X}, \mathbf{X})^{-1}k(\mathbf{X}, \mathbf{x}^*)) \end{aligned} \quad (54)$$

Figure 3 shows prior and posterior samples from a GP, as well as contours of the predictive density. Our use of probabilities does not mean that we are assuming the function being learned is stochastic or random in any way; it is due to probabilistic inference incorporated in the method.

The flexibility of GP models raises the question of which kernel to use for a given problem. Choosing a useful kernel is equivalent to learning a useful representation of the input. Kernel parameters can be set automatically by maximizing the marginal likelihood. Although there are attempts to make the process automated [18], it may be essential to have a human domain expert to choose the parametric form of the kernel.

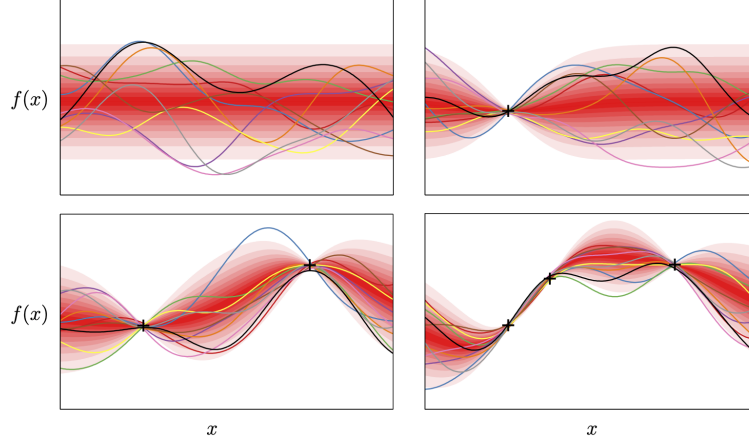


Figure 3: A visual representation of a Gaussian process modeling a one-dimensional function. Different shades of red correspond to deciles of the predictive density at each input location. Coloured lines show samples from the process – examples of some of the hypotheses included in the model. Top left: A GP not conditioned on any data-points. Remaining plots: The posterior after conditioning on different amounts of data. All plots have the same axes. The Figure taken directly from [18].

### Kernel construction

Let us examine sampled functions encoded by some commonly used kernels: the squared-exponential (SE), periodic (Per), and linear (Lin) kernels. These kernels are defined in Figure 4.

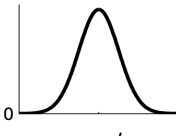
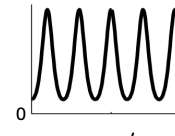
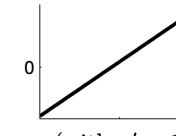
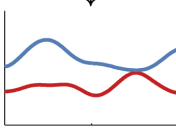
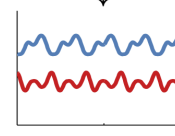
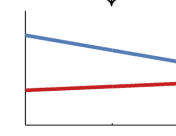
Kernel name:	Squared-exp (SE)	Periodic (Per)	Linear (Lin)
$k(x, x') =$	$\sigma_f^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$	$\sigma_f^2 \exp\left(-\frac{2}{\ell^2} \sin^2\left(\pi \frac{x-x'}{p}\right)\right)$	$\sigma_f^2 (x-c)(x'-c)$
Plot of $k(x, x')$ :			
	$x - x'$	$x - x'$	$x$ (with $x' = 1$ )
	↓	↓	↓
Functions $f(x)$ sampled from GP prior:			
	$x$	$x$	$x$
Type of structure:	local variation	repeating structure	linear functions

Figure 4: Examples of structures expressible by some basic kernels. The Figure taken directly from [18].

Each co-variance function corresponds to a different set of assumptions made

about the function we wish to model. For example, using a squared-exp (SE) kernel implies that the function we are modeling has infinitely many derivatives. There exist many variants of “local” kernels similar to the SE kernel, each encoding slightly different assumptions about the smoothness of the function being modeled.

Kernels can be added and/or multiplied, thus gaining new properties and enabling modelling of data with higher complexity.

**Kernel hyper-parameters:** Each kernel has a number of parameters which as also referred to as hyper-parameters, since they can be viewed as specifying a distribution over function parameters, instead of being parameters which specify a function directly. An example would be the length-scale parameter  $l$  of the SE kernel, which specifies the width of the kernel and thereby the smoothness of the functions in the model.

## 5 Financial data analysis

Financial data analysis is centered around an idea that through mathematical modeling of historical data, one can predict (to a certain degree) future prices of financial derivatives. However, the assumption that it is possible is against the efficient market hypothesis (EMH) [19] which since its formulation in 1970, is still an active subject of discussions among financial professionals. [20] EMH states that asset prices reflect all available information. If the hypothesis is true, then only insiders with information not available to the public could beat the market by utilisation of temporal mispricings.

As a result of the prices fully reflecting all known information, even uninformed investors buying a diversified set of financial derivatives such as stocks, at the market prices, would obtain return as good as that achieved by the experts.

Existence of trading companies led by mathematical researchers, that consistently deliver market beating results [21] can serve as an empirical evidence against the efficient market hypothesis, and be a driving force behind financial data research.

In this chapter, we will discuss an example of historical market inefficiency and briefly discuss the assumption of stationarity of financial data.

### 5.1 Market efficiency

Once people start to trade against certain patterns, the patterns tend to vanish. This is because our trades influence the market — especially when there are significant volumes involved. An example of a vanished pattern is the trending of many commodities in the 1980s related to seasons - see Figure 5 for the chart. One could buy when the price is above a certain moving average and sell when the price is below it, thus realising profit based on the seasonal effect.

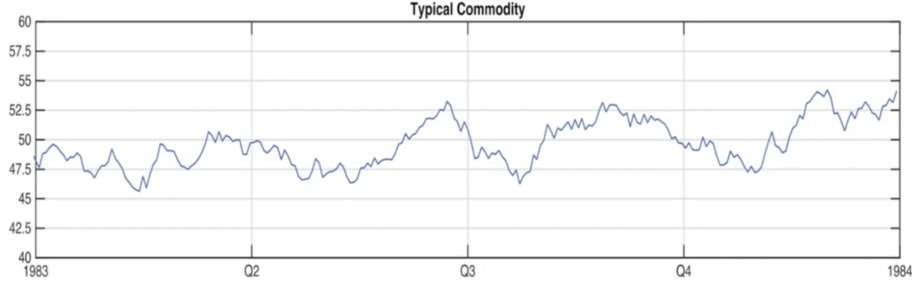


Figure 5: Price of a typical commodity (sugar) in 1984 with visible seasonal effects which many commodities shared at that time. A primitive system of trend following could give an investor an advantage over the market. Nowadays price patterns are much more complicated. [22]

Groups of people behave in patterns, and investors are no different. It should be possible to advantage of people being somewhat predictable with the right mathematical model, which needs to evolve constantly, just like that group of people.

## 5.2 Stationary of the market dynamics

Despite the ever-changing geopolitical environment, in the finance literature, it is common to assume that the distributions (and therefore the dynamics) governing time-series of percentage-wise changes are stationary. It can be checked empirically provided that we gather sufficient number of historical prices. For example, one can divide the data into sub-samples and check the consistency of the results obtained across the sub-samples.

# 6 Experimental data and feature extraction

This section marks the end of the theoretical part of the thesis, and the start of the experimental part where we focus on the actual data and modeling results. We will list the necessary details regarding the financial data and address modeling problems that can be reduced or solved using appropriate feature extraction.

## 6.1 Data sources and normalisation

We based our main experiment on BTC/USDT (Bitcoin to Dollar) transaction data from the start of 2018 until the end of 2021 - recorded on the Binance exchange. A resulting time-series is illustrated in Figure 6.

The dataset consists of over 80 million recorded transactions. We made part of our aggregated data public (prices and transaction volumes). It is available (1 min aggregation) [here](#).



Figure 6: Bitcoin prices for train and test&validation datasets. Vast differences between data distributions between these two regions which has to be addressed by proper feature extraction/normalisation.

### Normalisation

Normalisation is a technique often applied as part of data preparation for machine learning. The subject is broadly discussed in [23]. The goal of normalisation is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information. Normalisation is meant to remove long term memory effects in the system, thus easing exchange of information between different parts of the training dataset.

We transformed all prices to the  $[-1.5:1.5]$  range using the following formula:

$$y = \arctan \frac{p - m_1}{cm_2} \quad (55)$$

where  $y$  is the chosen normalised feature,  $p$  are the prices,  $m_1$  and  $m_2$  are short and long term moving averages of prices (3 hours and 4 days respectively), and the parameter  $c$  is learned from the training data such that 90'th percentile of the transformed features in the train dataset region is smaller than  $\frac{\pi}{6}$  and 10'th percentile is bigger than  $-\frac{\pi}{6}$ . The motivation behind the introduction of the parameter  $c$  is to force the transformation to be semi-linear, because the DMD algorithm utilises the Frobenius norm. Other normalisation methods such as log differences between consecutive prices are heavily nonlinear, as small deviation in them would result in large deviations in prices. This constrain can be rephrased as looking for a mapping which is locally linear. We can inspect the results illustrated in Figure 8, where we see that the feature mapping looks locally (within a potential input length) linear, despite being nonlinear globally as illustrated in Figure 6.

### Stationarity of normalised data

Input features values distribution should be stationary - we already discuss its importance in Section 6.1. Here we want to assess the degree of price data distribution stationarity after the normalisation equation (55). We divide training dataset into 3 regions of equal length of 8 months and illustrate the value distributions in Figure 7.

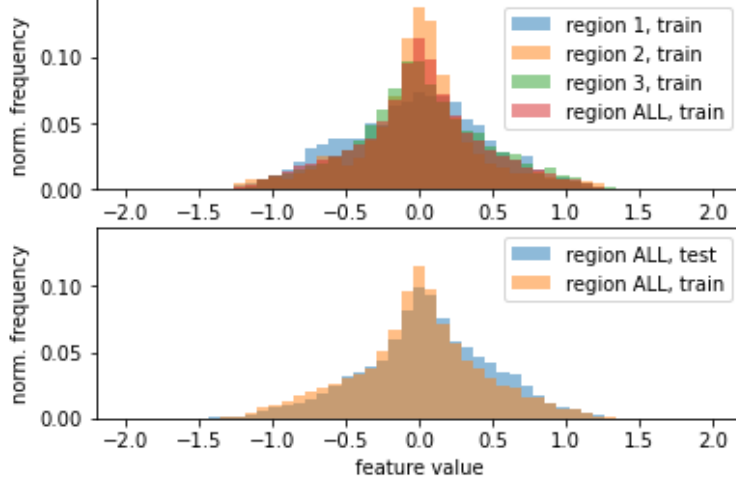


Figure 7: Histogram of feature values (prices after normalisation). Top: the train dataset divided into 3 regions of equal length of 8 months, and combined. Bottom: train and test. Data distributions share a similar structure, but would fail statistical tests such as Kolmogorov–Smirnov test for stationarity.

The distributions ultimately fail the Kolmogorov–Smirnov test for stationarity, but it is unclear if there exists a transformation to have near identical value distributions while being locally linear.

## 6.2 Price trajectories

Before we feed in the transactions into any of our models, we aggregate transactions further into price trajectories (short arrays of aggregated prices). Each price trajectory represents many time-binned data-points, thus potentially containing enough information to contain information with forecasting capabilities.

In DMD modelling after coordinates transformation, we restrict our-selves only to selected, simple dynamics. To capture non-linearity of our system we want to expand dimensionality of our features. We transform each price trajectory post normalisation (a 1D array) into a 2D array of pixels, where each pixel is either 0 or 1, depending on the price value at a given time window. Further in the text,

## 6 Experimental data and feature extraction

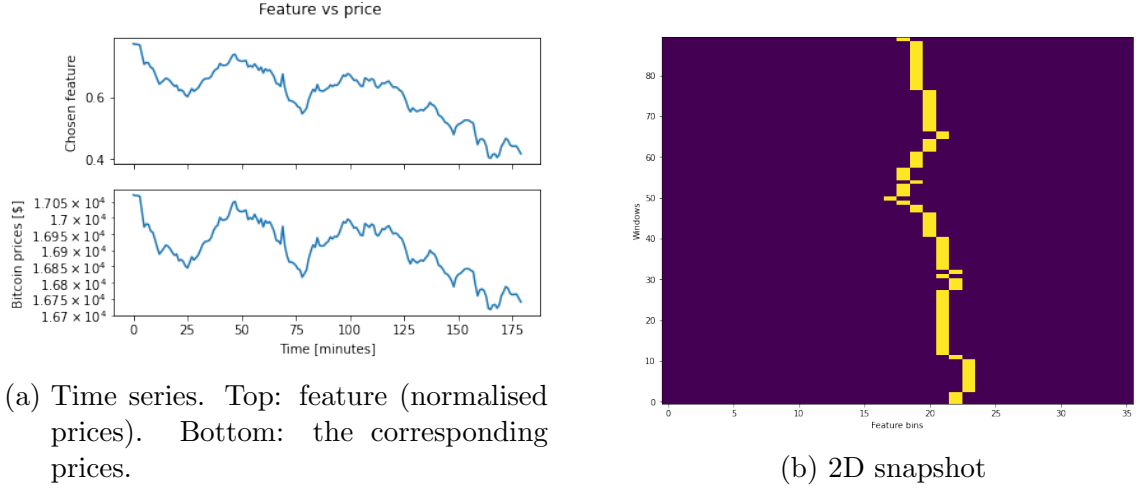


Figure 8: Price & feature time series (values vs time), and the corresponding 2D snapshot (time windows vs feature bins) .

our transformed snapshots will be interpreted as 2D PDFs (method also introduced in [6]). For true prices (not forecasts) we know the positions with certainty. For forecasts, the pixels can be between 0 and 1 depending on the confidence of the model and interpreted as a given probability.

An illustration of an example price trajectory snapshot is in Figure 8.

### 6.3 Comparison metrics

Here we list our metrics of interest for 2D snapshots comparison and later introduced label classification.

#### Metrics of interest for 2D snapshots:

- PDF Entropy / Uniform PDF Entropy
- Intersection over Union (IoU) - sum over PDF pixels along a true price path, divided by the number of time windows
- Path probability ( $-Log_{10}P_{pp}$ ) - logarithm of a product of PDF pixels along a true price path

#### Metrics of interest for labels:

- confusion matrix to assess accuracy

To calculate entropy of a given 2D snapshot (2D PDF) we use the standard formula from information theory:

$$H = \sum_i P(p_i) \log P(p_i) \quad (56)$$

where  $H$  is the entropy and  $p_i$  is a value of a particular pixel. The sum iterates over



all pixels in the 2D snapshot.

#### 6.4 Goodness of selected 2D snapshot resolution - mismatch index

Here we want to focus on a problem of choosing the right 2D snapshot resolution: the appropriate number of feature bins and time windows. One can speculate if it is even theoretically possible to reliable forecasts prices, given the discussion regarding market efficiency in Section 5.1. We want to have an answer to this question. In example, how challenging it would be to accurately forecast 100 time windows into the future, given the fact that we use past 200 time windows as the input ? The subject is covered already in [24] and [25], however the proposed method works for supervised problems and can not be easily applied to unsupervised learning. We set the number of feature bins to 36, given the fact that it is used in [25]. The number of time windows and predictability time-horizon remains a question. We allow the number of time windows in input 2D snapshots and the effective number of time windows in output 2D snapshots to differ. By effective, we mean the fact that by DMD construction they have to be the same, but we can disregard part of the output from evaluations. To limit the computational challenge of searching the right parameters, we restrict time-window lengths to be 3 minutes, however the number of them in a snapshot can vary.

Based on discussions in [24] and [25], we invent a new index, which we call *mismatch* index (or *mmIndex* for short) in the following way: we calculate Root Mean Square (RMS) differences between each two input 2D snapshots. This RMS serves as a measure of distance between two input 2D snapshots. We do the same thing for each two output 2D snapshots. By proper inspection of these distances, we can calculate how often a relatively close (RMS distance-wise) input snapshot pair results in the respective output 2D snapshots to be far-away. Ideally, close inputs should results in close output, because otherwise our model has to exhibit a higher degree of non-linearity. Worst case for modeling scenario, identical inputs results in different outputs which disallow accurate forecasting even theoretically. If the input distance is smaller than the 20'th percentile of input distances array, and the corresponding outputs are further away than the 80'th percentile of the output distances array, such even is classified as a mismatch. Mismatch index is a ratio of mismatch events to all possible events. If *mmIndex* is low, there should exist a model to accurately forecasts the future. The lower the *mmIndex* the better the potential model.

The inverse of *mmIndex* can be seen as a measure of goodness of selected representation of resolutions/features - treated as a method agnostic selection criteria for the 2D snapshot.

To better communicate the idea behind the mismatch index, in Figure 9 we present a 2D heatmap of *mmIndex* values given particular input and output time-window numbers.

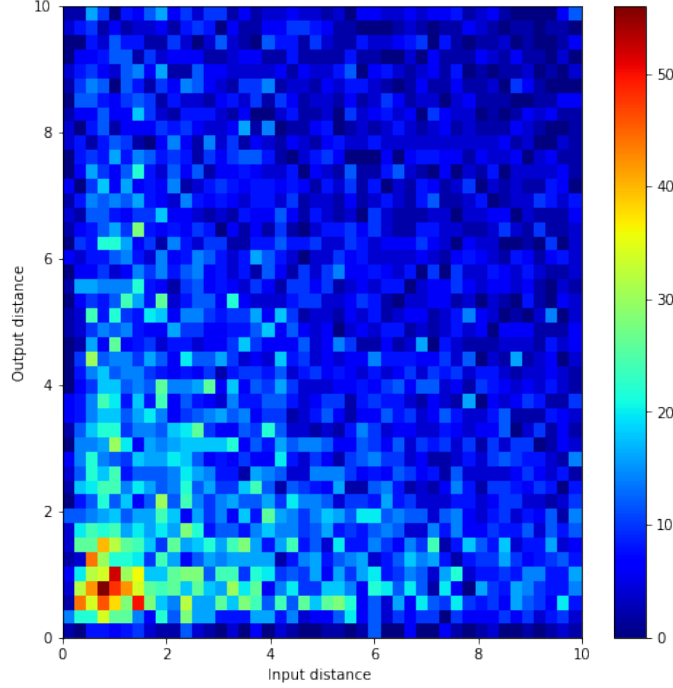


Figure 9: 2D heatmap of mmIndex values for input length 75 and output length 75. 20th percentile of input distances: 2.17, 80th percentile of output distances: 8.6, mmIndex: 7.7

Now we want to construct a 2D heatmap of mmIndex values for different input/output lengths. The result is presented in Figure 10

We observe that depending on the number of time windows in input/output 2D snapshots, the mmIndex can change up to 3 times the found minimal value. To supplement our mmIndex analysis, we calculate average IoU values for each particular input/output length and each method (DMD, GP, RW). The results are illustrated in Figures 11, 12, and 13. More about IoU values and their distributions in Section 9.

We calculate linear correlations across all tested lengths between IoU and mmIndex:

- mmIndex vs DMD: -0.487
- mmIndex vs RW: -0.608
- mmIndex vs GP: -0.713

Based on these correlation values, further experiments will be conducted using 75 time windows in the input and 50 time windows in the output (mmIndex 4.9). Only one configuration (input 75, output 25) results in lower mmIndex equal to 4.7, but this configuration would result in 33% more snapshots and push already high computational complexity of modeling even further. The bottle neck of the experiments is the GP model which RAM memory usage scales as a square of the

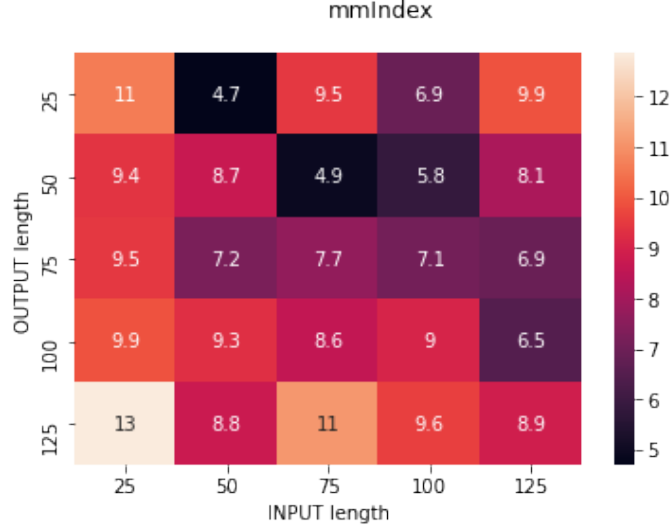


Figure 10: 2D heatmap illustrating mmIndex values (lower is better) for various input/output lengths of 2D snapshots. Depending on the 2D snapshots resolution, mmIndex can change up to 3 times the found minimal value.

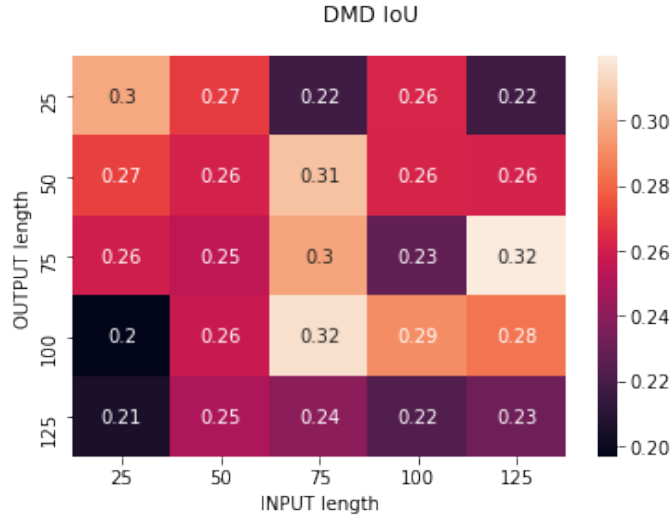


Figure 11: 2D heatmap illustrating DMD IoU (higher is better) for various input/output lengths in 2D Snapshots.

number of inputs.

We complete this subsection by illustrating in Figure 14 a cross section of Figures 11, 12, and 13 for input length equal 125 time windows. We see observe the previously mentioned clearly negative linear correlations.

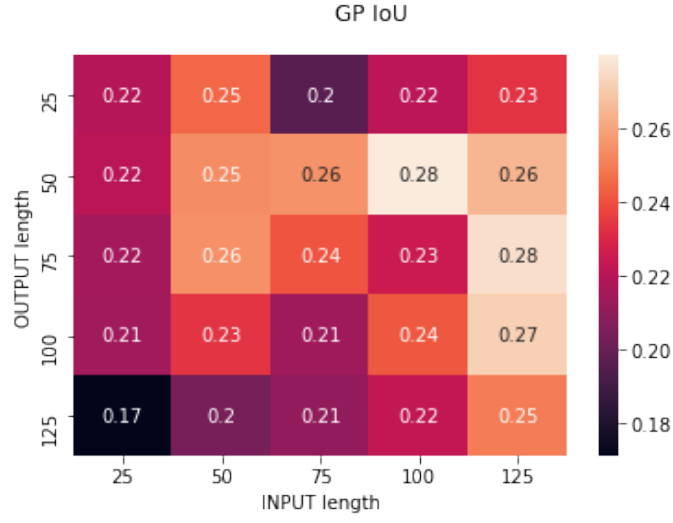


Figure 12: 2D heatmap illustrating GP IoU (higher is better) for various input/output lengths in 2D Snapshots.

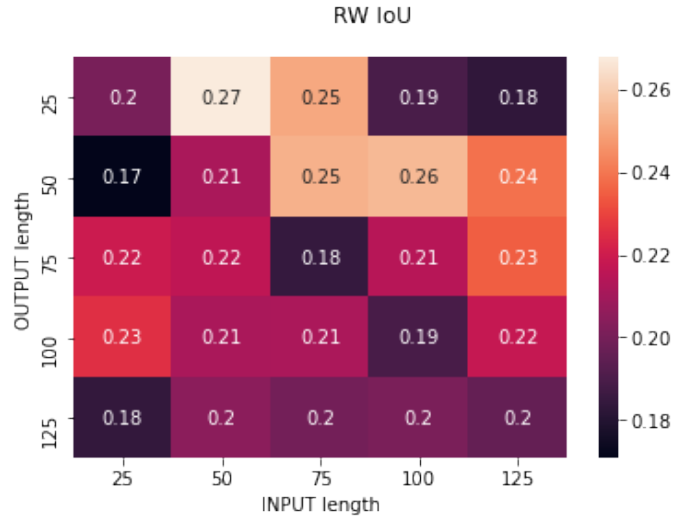


Figure 13: 2D heatmap illustrating RW IoU (higher is better) for various input/output lengths in 2D Snapshots.

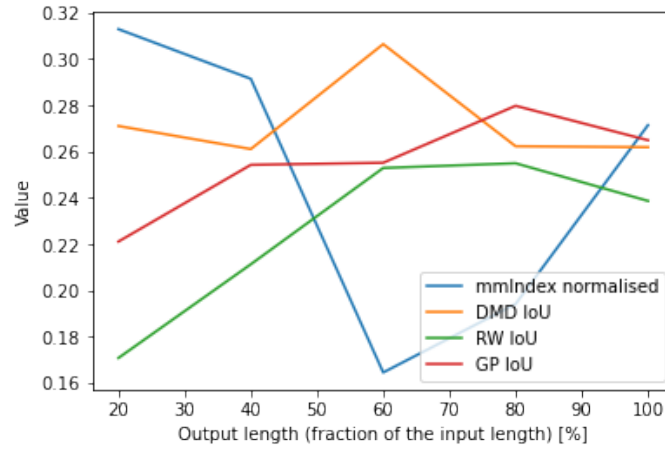


Figure 14: illustration of mmIndex and IoU values vs different input/output data fractions. The figure suggest that the predictability power both theoretical (mmIndex) and achieved (IoU) are best when we forecast outputs which lengths are around a half of the input lengths. mmIndex anti-correlates with IoUs.

## 6.5 Data pipeline

Here we summarise the data section with an illustration of the data pipeline in Figure 15.

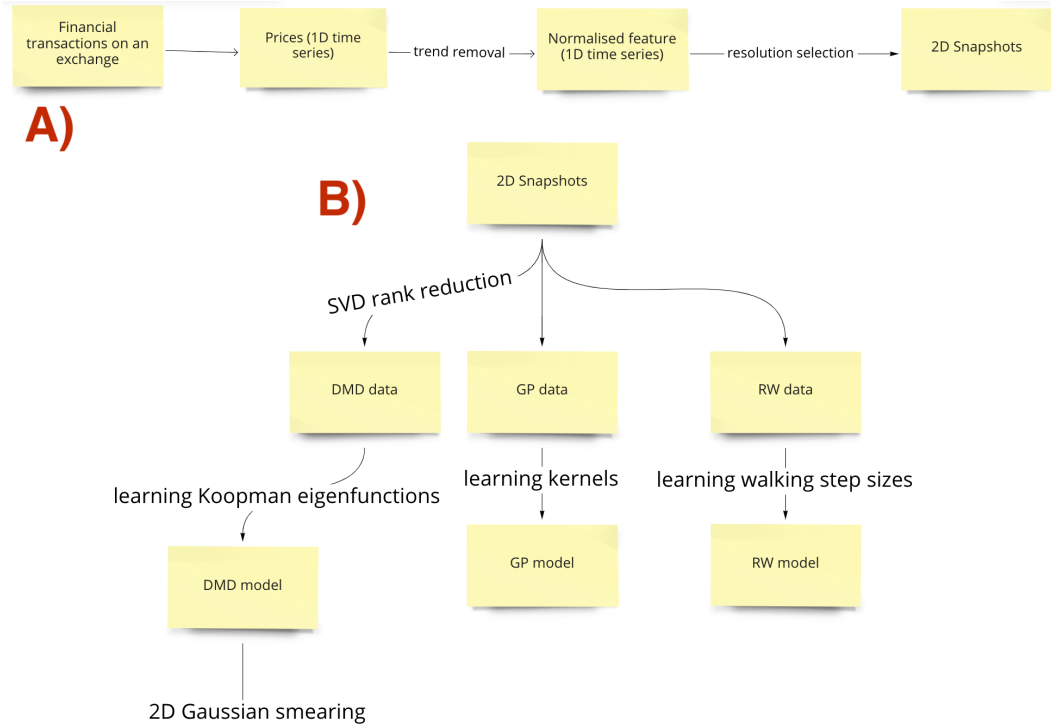


Figure 15: Two pipelines. A) Data preparation pipeline from raw transactions to 2D Snapshots - model inputs. B) Model building pipeline. From inputs to models. We utilise SVD rank reduction only for DMD data as a regularisation method. GP and RW have significantly less free weights/-parameters.

More about the B) part of Figure 15 in the following Section 7.

## 7 Hyper-parameters optimisation pipeline

In this section we discuss the variety of parameters that have to be find or learn in order to utilise DMD, GP, and RW modeling effectively.

### 7.1 Common parameters

Common parameters relate to input representation and should be model agnostic - not to favour a particular model. To find appropriate 2D snapshot resolution we utilise mmIndex (introduced in Section 6.4) as a loss function.

## 7.2 Model specific parameters

### DMD

We have to learn the regularisation strength - appropriate SVD rank truncation, which is a single parameter. In addition, as introduced in Figure 15 we have to learn the strength of 2D Gaussian smearing, which is a standard deviation of the internal Gaussian filter. The smearing is beneficial for the forecasting as it solves two problems. First of all, some of the values of the DMD model output (pixels in a 2D Snapshot) can be slightly negative. We replace them with zeros which would result in path probability equal to zero in some edge cases. In addition, we would want a small deviation in price to result in a small deviation in prediction. Both RW and GP models have these properties, as explained in subsections below, they do have noise terms.

### GP

Utilising our domain-specific knowledge, we have decided to define the kernel as a weighted sum of squared-exponential and periodic kernels (introduced in Figure 4). In addition, we have included a white kernel (noise term), which is defined as follows:

$$k(x, y) = \begin{cases} 0 & x = y \\ c & x \neq y \end{cases} \quad (57)$$

where constant term  $c$  has to be learned from data.

### RW

The only parameter we have to learn is the standard deviation of the input 2D snapshot, which can be computed effortlessly. Last seen price entry serves as the mean value for the output distribution.

## 7.3 Search algorithm

To efficiently search the hyper-parameters space we rely on Bayesian Optimisation [26], Hyper-Band [27], and k-fold cross-validation. The goal of each search is to maximise score (average IoU) or loss (mmIndex) in a given validation region.

## 8 Experimental results: sine wave with noise

Before we tackle high-complexity modeling challenges, we want to build modeling intuition on a much simpler example. In this section we will model sine wave with

noise. The noise we choose to be Gaussian of zero mean and standard deviation such that  $\frac{\sigma_{\text{sine}}}{\sigma_{\text{noise}}}$  is equal to 3. We want to see how our feature extraction and DMD learning algorithm will be able to forecast such data. The resulting time series is illustrated in Figure 16.

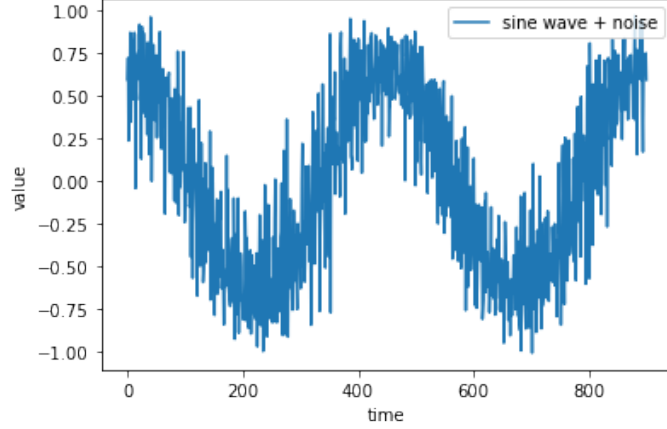


Figure 16: sine wave with noise represented as a time series. The standard deviation ratio:  $\frac{\sigma_{\text{sine}}}{\sigma_{\text{noise}}}$  is equal to 3.

## 8.1 Data processing

To our sine wave with noise data we now apply the processing pipeline previously illustrated in Figure 15 section A. As a result of this processing, we have 2D snapshots illustrated in Figure 17

We now apply SVD decomposition (introduced in Section 3.2) to further explore the data structure. Given the "rotational" dynamics between 5 different stages illustrated in previously mentioned Figure 17, we should expect first 5 singular values to be much higher than the other. The relevant plot is illustrated in Figure 18.

## 8.2 DMD modeling

Here we take a look at the leading DMD modes and eigenvalue distributions for various SVD ranks of data. Figure 19 illustrates two leading DMD modes eigenvalue module-wise. Real part plots resemble smeared straight line trajectories. The eigenvalue distributions plots illustrated in Figures 20, 21 reveal interesting structure of eigenvalues. For smaller ranks, the leading five eigenvalues with  $|\lambda| \approx 1$  correspond to system dynamics, the rest results in overfitting the noise and form a growing uniform circle in the eigenvalues distribution plot. For larger ranks, the separation between eigenvalues corresponding to system dynamics and overfitting from Figure



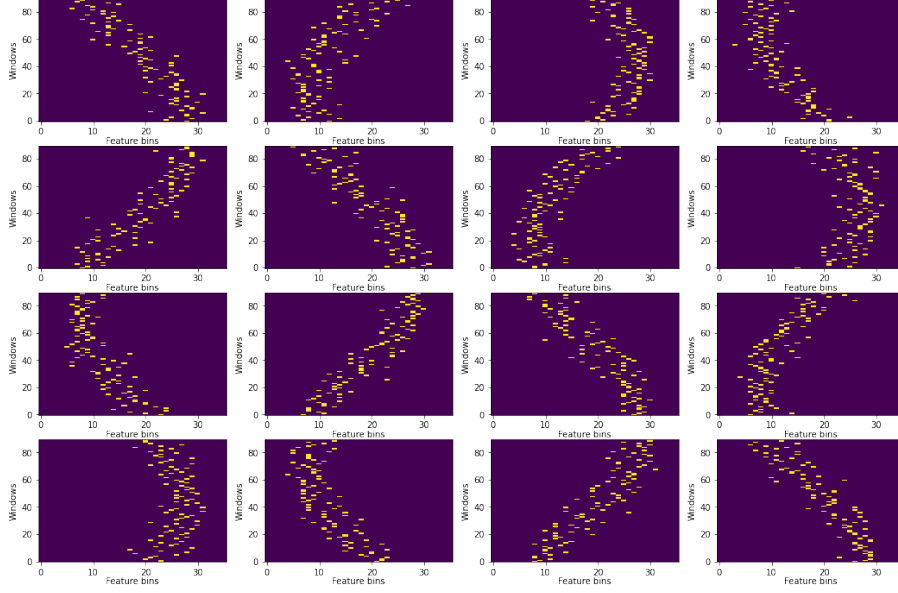


Figure 17: Sine wave with noise represented as 2D Snapshots. To get the chronological order of snapshots one should read the figure from left to right and then down. Each yellow pixel represents a binned price position bin-wise (X-axis) at a given time window (Y-axis). The period of the sine is 5 times longer than the time-wise resolution of the 2D Snapshots, resulting in "rotational" dynamics between 5 different stages.

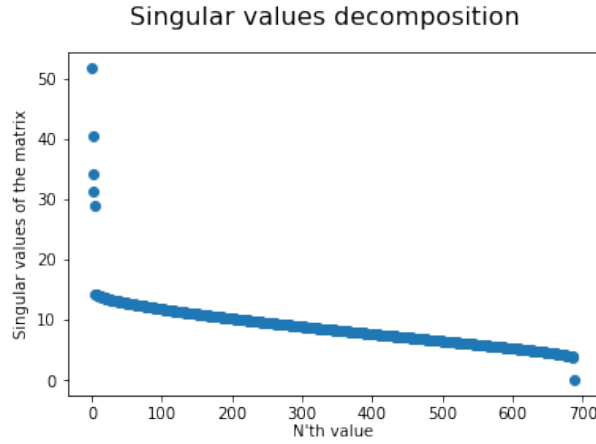


Figure 18: Singular values - results of SVD decomposition of sine with noise data. The highest 5 values (first 5 values) are at least the double of the rest of values. It is connected to the fact that the the period of the sine is 5 times longer than the time-wise resolution of the 2D Snapshots, resulting in "rotational" dynamics between 5 different stages.

20 is no longer visible. The overfitting eigenvalues stabilised and form a ring with  $|\lambda| \approx 1$ .

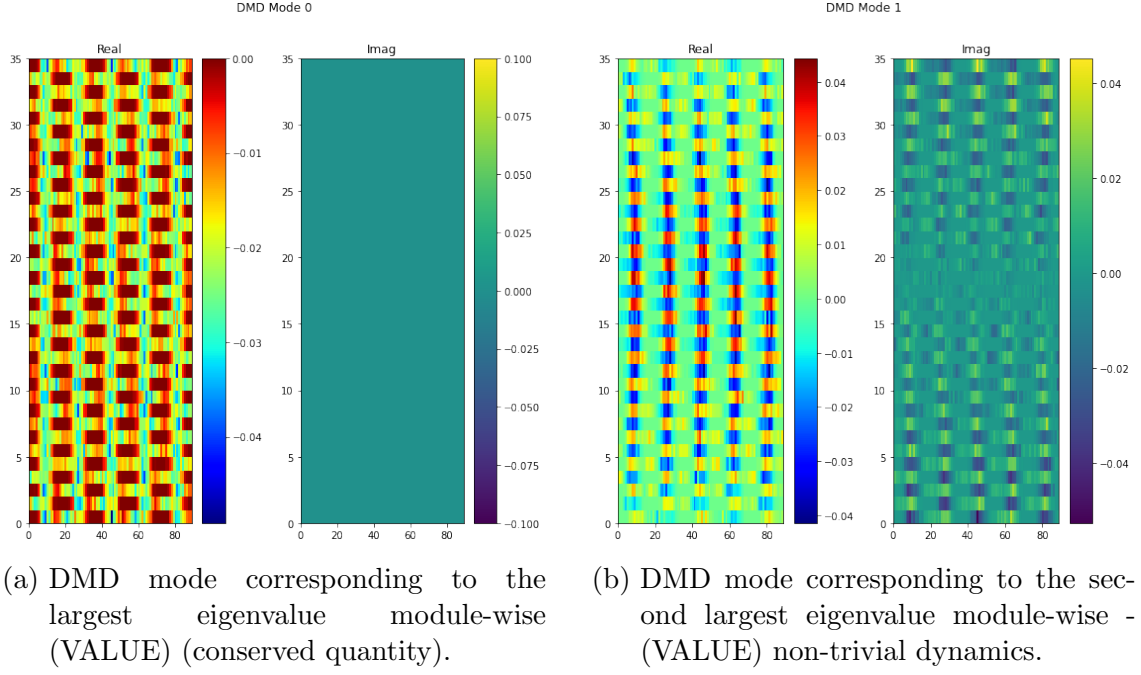


Figure 19: Two DMD modes corresponding to two largest module-wise DMD eigenvalues. Real part plots resemble smeared straight line trajectories.

To illustrate the overfitting when  $r > 5$ , we present Figure 22 which contains average IoU values for train vs test datasets.

One can wonder what kind of operator can be constructed using only the overfitting eigenvalues/eigenvectors - identifying its properties could help with identifying overfitting. We construct operator  $\mathbf{B}$  in the following way:  $\mathbf{B} = \mathbf{A}_{r>5} = \mathbf{A} - \mathbf{A}_{r=5}$ , where  $\mathbf{A}$  is the DMD operator introduced in Section 2.3 and its subscripts indicate which SVD projections we perform. Histogram of values of such an operator shares close resemblance with a Gaussian distribution, which links the matrix of the  $\mathbf{B}$  operator to random matrixes which coincidentally also have uniform circles in its eigenvalue distributions - histogram illustrated in Figure 23.

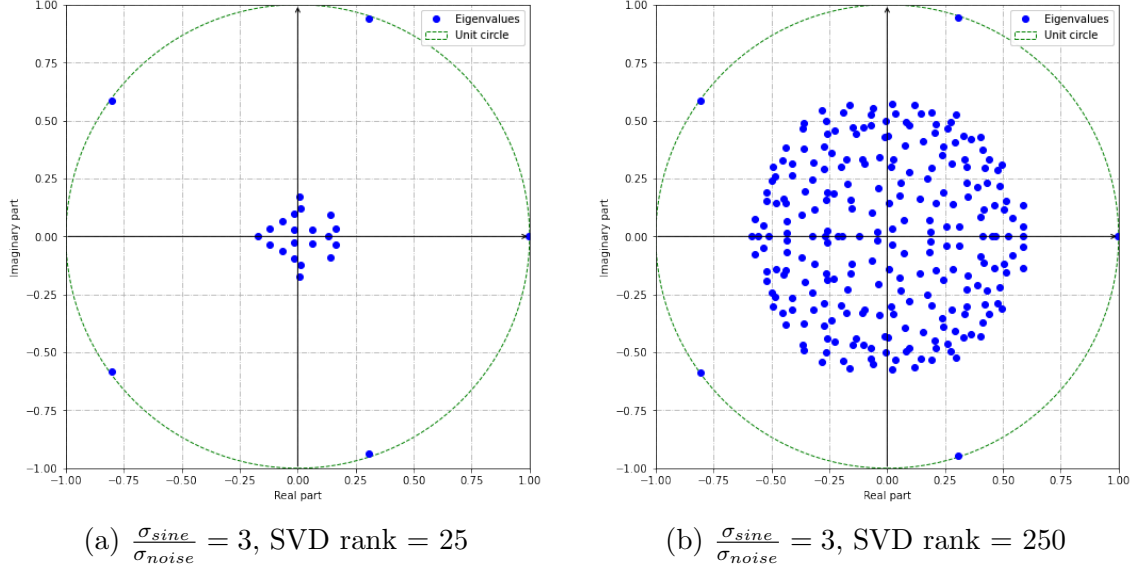


Figure 20: Distribution of eigenvalues of the DMD model for sine wave with noise  $\frac{\sigma_{sine}}{\sigma_{noise}} = 3$ , (a) SVD rank = 25 and (b) SVD rank = 250. Five eigenvalues with  $|\lambda| \approx 1$  correspond to system dynamics, the rest results in overfitting the noise and form a growing uniform circle in the eigenvalues distribution plot.

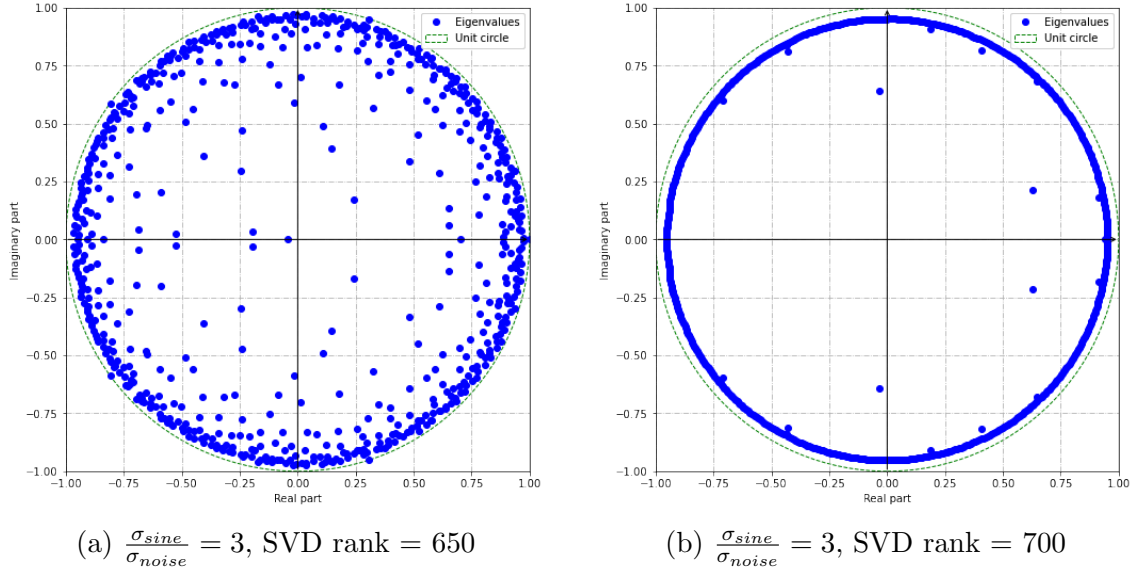


Figure 21: Distribution of eigenvalues of the DMD model for sine wave with noise  $\frac{\sigma_{sine}}{\sigma_{noise}} = 3$ , (a) SVD rank = 650 and (b) SVD rank = 700. Separation between eigenvalues corresponding to system dynamics and overfitting from Figure 20 is no longer visible. The overfitting eigenvalues stabilised and form a ring with  $|\lambda| \approx 1$ .

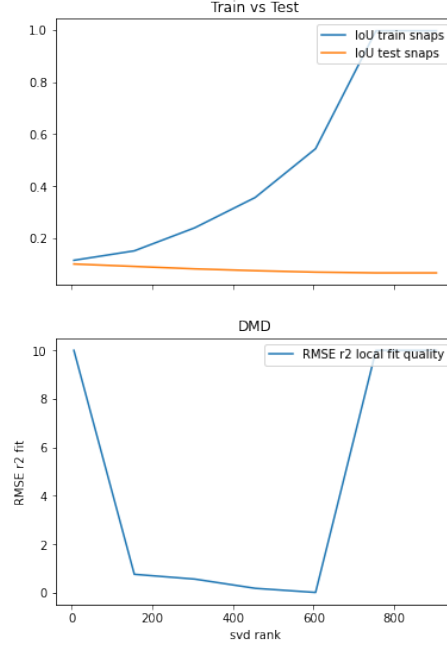


Figure 22: From the top: a) IoU score (higher is better) of sine with noise for different SVD ranks and b) root mean square error (RMSE) between a fitted quadratic function  $f(r) = r^2$  and function  $g(r) = \sum_{|\lambda| < r} \lambda_i$ . It illustrates the uniform circle formation in the eigenvalue distribution of the sine with noise. Low RMSE values indicate high fit quality - indication of a clear uniform circle between low and high SVD ranks (overfitting but with unstable eigenvalues that fade away).

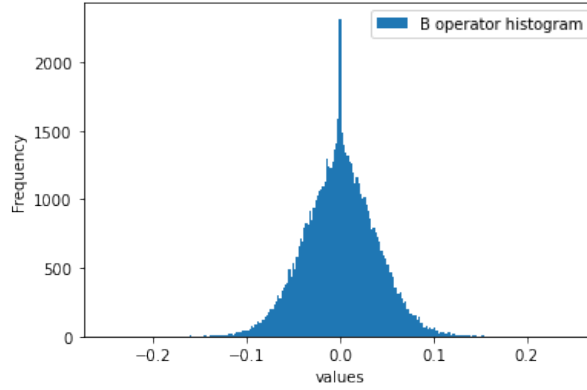


Figure 23: Histogram of value entries in the  $B = A_{r>5} = A - A_{r=5}$  matrix. SVD rank 250.  $A$  is the DMD matrix and  $A_{r=5}$  is a matrix build using only the first 5 highest module-wise eigenvalues. The resulting  $A_{r>5}$  has a uniform circle in the eigenvalue distribution which suggests it is a random matrix with Gaussian entries. The histogram confirms it.

### 8.3 2D snapshots reconstruction

In this subsection we present a series of figures (Figure 24, 25, 26) to illustrate reconstruction (everything within training dataset) capabilities of DMD 2D snapshot for various SVD ranks. From Figure 22 we already know that the best generalisation on the test dataset is for  $r = 5$  resulting in average IoU equal to 0.09, but here we want look into the degree of memorisation of training dataset for high values of  $r$ . Details regarding particular reconstructions are in the appropriate figure captions.

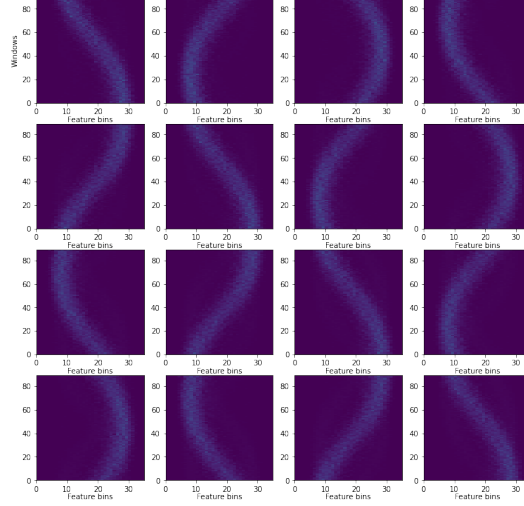


Figure 24: Sine wave with noise forested using DMD as 2D Snapshots. Reconstruction of Figure 17 data. SVD rank 5. The system does not reconstruct/memorise the noise (no overfitting).

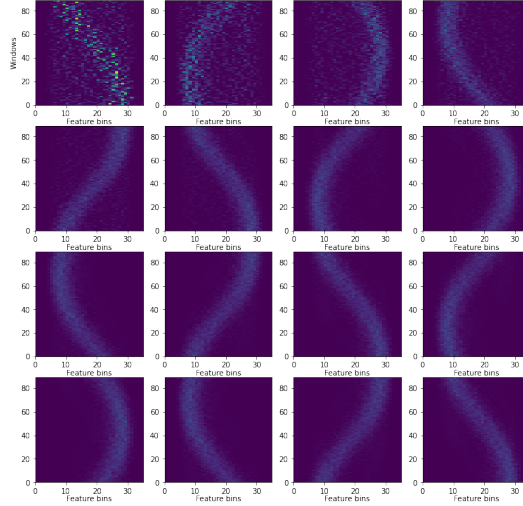


Figure 25: Sine wave with noise forested using DMD as 2D Snapshots. Reconstruction of Figure 17 data. SVD rank 250. The system does reconstruct/memorise the noise but only for the first few snapshots and only to a certain degree. The noise fades away as the noise uniform circle eigenvalues are not stable ( $|\lambda| < 1$ ).

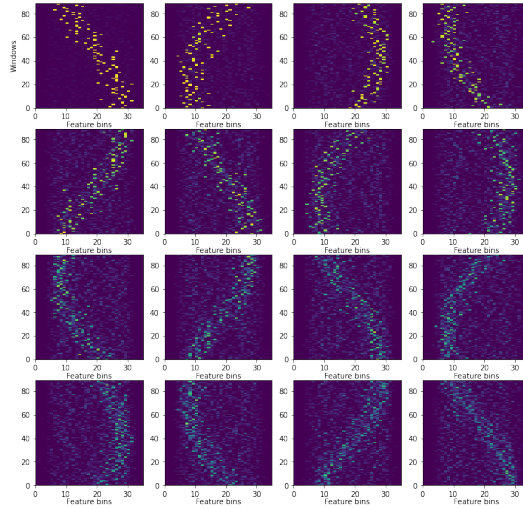


Figure 26: Sine wave with noise forested using DMD as 2D Snapshots. Reconstruction of Figure 17 data. SVD rank 650. The system does reconstruct/memorise the noise for many snapshots ahead. The noise fades slower than the one presented in Figure 25 as the noise uniform circle eigenvalues is now bigger (Figure 20 vs 21)

## 9 Experimental results: financial transactions

This section covers main results of the thesis - from data processing of financial transactions into 2D snapshots, forecasting 2D snapshots using DMD, GP, and RW, to classification of specialised financial labels and forecasting investment risks.

### 9.1 Data processing

To the financial datasets introduced in Section 6, we now apply the processing pipeline previously illustrated in Figure 15 section A. As a result of this processing, we have 2D snapshots illustrated in Figure 27

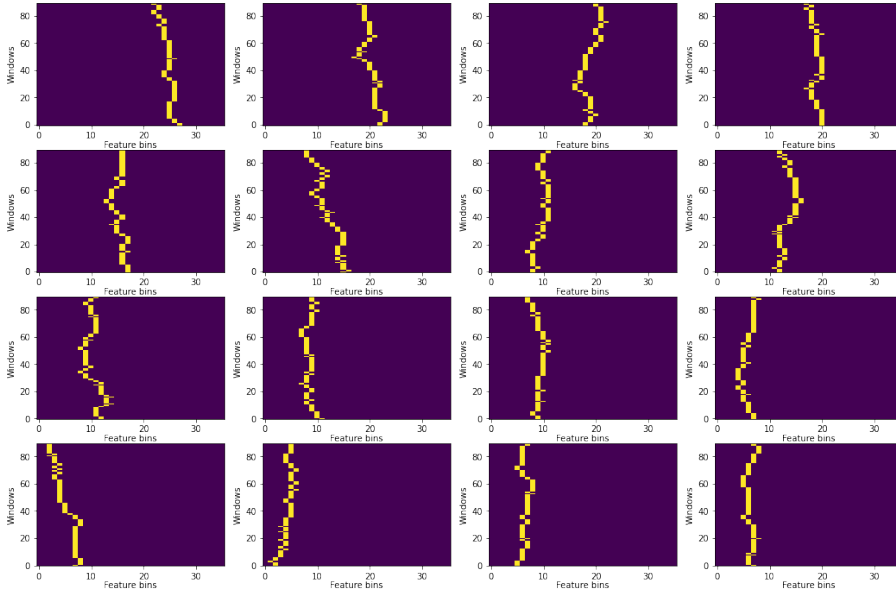


Figure 27: 2D Snapshots - price trajectories.

We now apply SVD decomposition (introduced in Section 3.2) to further explore the data structure. The relevant plot of singular values is illustrated in Figure 28.

SVD rank 1125 corresponds to 92% of area under the curve in the singular values decomposition (Figure 28), and was chosen during DMD specific hyper-parameter search (discussed in Section 7)

Influence of SVD rank on DMD modelling is illustrated in Section 8 concerning sine wave with noise. One could make analogies to the number of epochs in machine learning - too many might overfit the model, despite constant number of weights - learning training data by heart, instead of looking for patterns which can generalise. Fortunately, here in Figure 28 we observe a low-rank structure which does require relatively few singular values to represent the data, thus reducing the risk of overfitting.

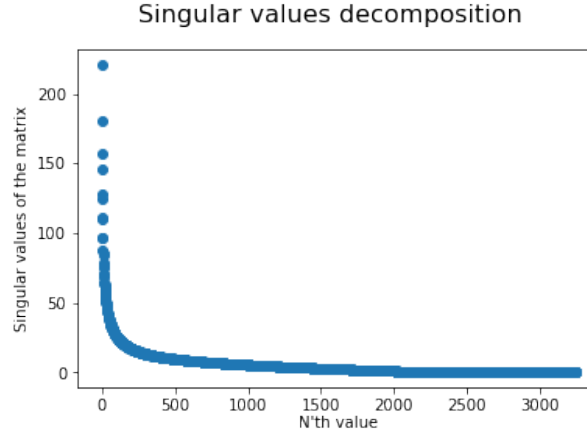


Figure 28: Singular values - results of SVD decomposition of price trajectories data. Visible existence of a low-rank structure which enables a data matrix rank cut.

## 9.2 DMD modeling

Here we inspect the DMD operator eigenvalue distribution. The structure visible in Figure 29 resembles the structure from Section 8.2. We observe a uniform circle and certain amount of satellites. Unfortunately, finding an appropriate SVD rank is not as easy as decreasing the rank until the uniform circle disappears. In such a scenario we remove too much information from the data and the model is unable to perform well. To find the appropriate SVD rank we employ hyper-parameter search introduced in Section 7. It results with SVD rank equal to 1125, which corresponds to 92% of area under the curve in the singular values decomposition plot (Figure 28).

To further inspect the DMD model, we look at the training dataset reconstruction process. We reconstruct/forecasts first 16 snapshots using the first one - illustrated in Figure 30. The reconstruction capability quickly fades away, which indicates even small forecasting horizon in the test dataset.



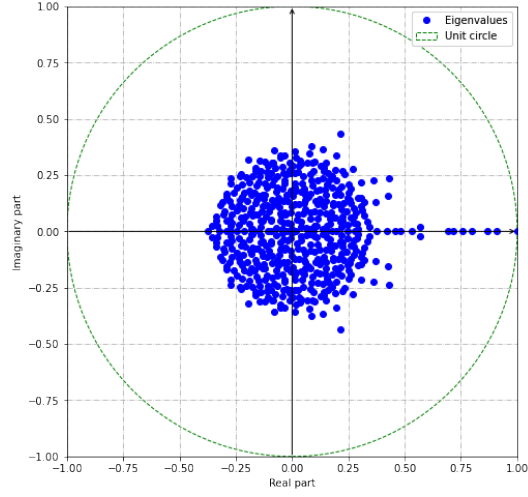


Figure 29: Distribution of eigenvalues of the DMD model for financial transactions. SVD rank 1125 corresponds to 92% of area under the curve in the singular values decomposition plot (Figure 28).

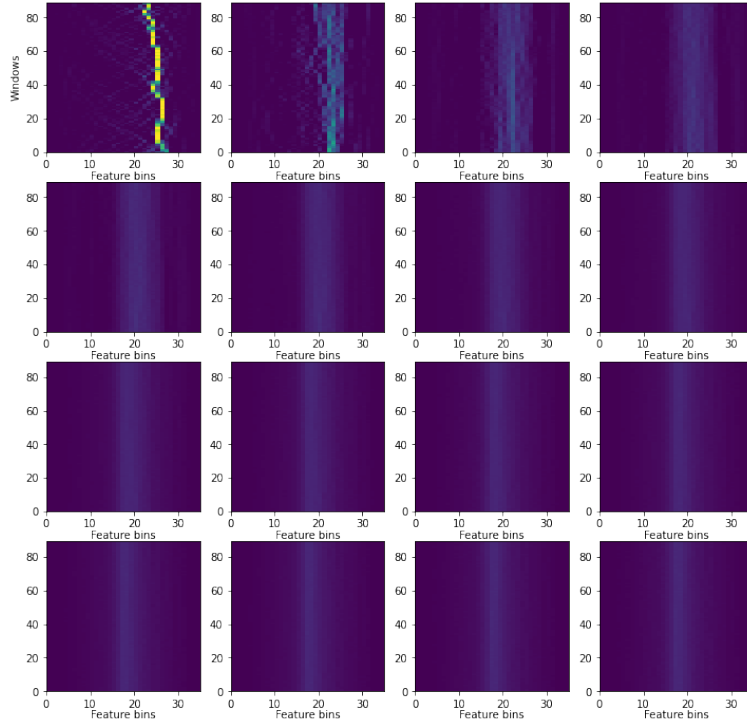


Figure 30: 2D Snapshots - price trajectories from Figure 27 reconstructed using the DMD. The forecast quickly fades away - small predictability horizon.

### 9.3 PDF trajectory forecasting

This subsection covers 2D snapshot forecasting. Firstly, us inspect Figure 31 to understand the structure of model outputs as 2D PDFs. In that particular output, we can observe path bifurcation in the PDF.

In addition, we present a particular forecast tackled by DMD, RW, and GP in Figure 32. We witness vast differences in structures of the forecasts depending on which method we use.

Figures 33, 34, 35 illustrate histograms covering IoU, entropy ratio, and price path probabilities for each method, further revealing differences between forecasting methods. The DMD has slightly higher IoU mean value:  $\mu_{DMD} = 0.311$  vs  $\mu_{GP} = 0.309$  and much smaller standard deviation:  $\sigma_{DMD} = 0.088$  vs  $\sigma_{GP} = 0.167$ . GP leads in price path probabilities. Which forecasting method between these two is better would depend on a use case. Same conclusion when it comes to comparing averages entropies - the degree of information a method broadcasts in a given forecast. Low entropy indicates high confidence in particular prediction. Entropy values for GP in this case are much smaller - the forecasts are more confident, but what is preferred might depend on the use case.

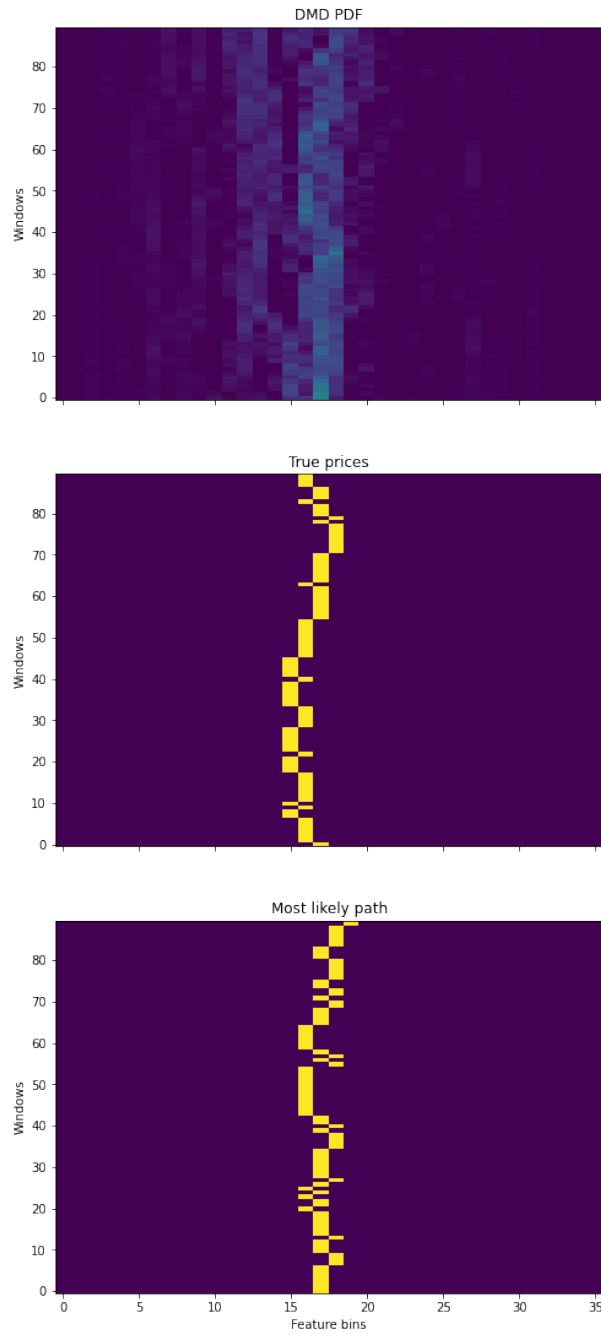


Figure 31: From the top: a) output of the DMD model, which can be interpreted as a 2D PDF, b) true prices - ground truth, c) most likely path derived from the 2D PDF

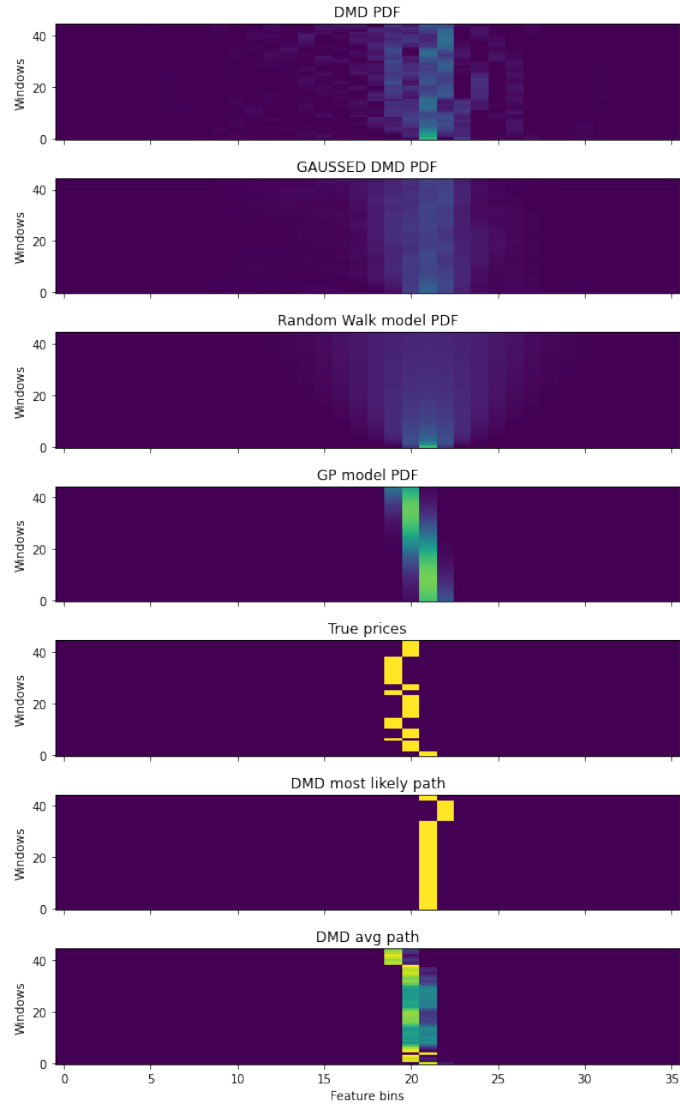


Figure 32: From the top: a) output of the DMD model, which can be interpreted as a 2D PDF, b) DMD output after Gaussian filtering, c) RW model output - forecast without a drift, d) GP model output, e) true prices - ground truth, f) most likely path derived from the 2D PDF, g) average path derived from the 2D PDF, where we allow pixels to be smeared if average value is fractional

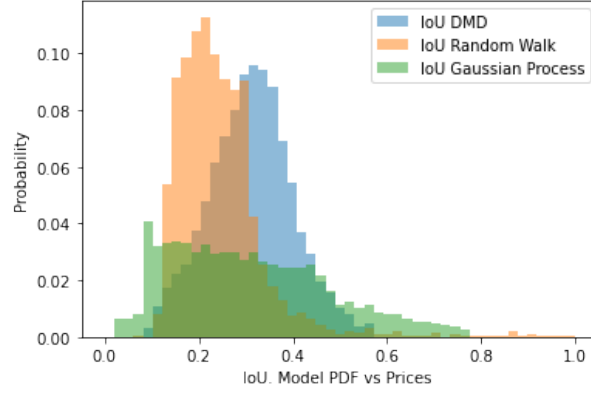


Figure 33: Histogram of IoU values for DMD, GP and RW.  $\mu_{DMD} = 0.311, \sigma_{DMD} = 0.088, \mu_{GP} = 0.309, \sigma_{GP} = 0.167, \mu_{RW} = 0.249, \sigma_{RW} = 0.116$

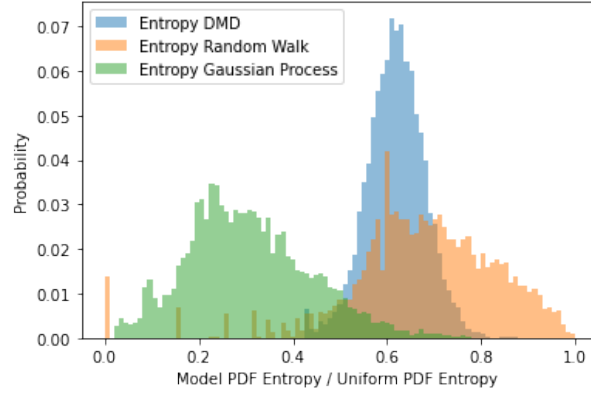


Figure 34: Histogram of entropy ratio values for DMD, GP and RW.  $\mu_{DMD} = 0.618, \sigma_{DMD} = 0.061, \mu_{GP} = 0.223, \sigma_{GP} = 0.341, \mu_{RW} = 0.591, \sigma_{RW} = 0.395$

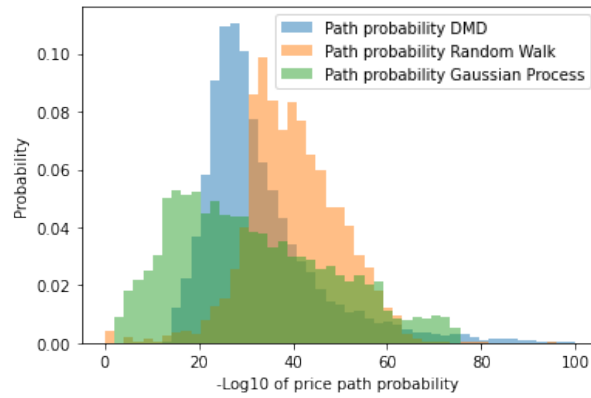


Figure 35: Histogram of PPP values for DMD, GP and RW.  $\mu_{DMD} = 32.1, \sigma_{DMD} = 8.1, \mu_{GP} = 31.7, \sigma_{GP} = 16.9, \mu_{RW} = 41.1, \sigma_{RW} = 9.2$

## 9.4 Financial labels forecasting

Every trade requires an exit, at some point. Establishing where to get out before a trade even takes place allows a risk/reward ratio to be calculated on the trade. In practice, a trader establishes a profit target - estimation of how far the price will move up. In addition, trades often involve a set target exit, often referred to as stop loss - estimation of intended risk on a particular investment/trade.

Metrics for 2D PDF comparisons can be misleading in financial applications, as discussed in [8] - we need to see if we can predict certain high-level characteristics of prices, which matter. Ability to predict high level features discussed in this section will also further differentiate DMD, RW and GP. RW is clearly worse at price forecasting but still is widely used - we will try to answer why it can still be useful (despite the obvious answer that it is much easier to build RW model since it requires only a single snapshot of data instead of thousands). Based on discussions in [8] and [7], we define a profitability label requiring two parameters: target profit (TP) and stop loss (SL). The formula for label is as follows: a) difference (in feature bins) between the last seen price in the input and the last price in the output if both TP and SL are not hit by the true price at any point, b) value of SL or TP otherwise, depending on which threshold was hit first. This label grades input&output pair depending on profitability of a potential trade.

We establish a one parameter ( $\alpha$ ) trading strategy as follows: buy the label classified by a method of use grades the profitability above  $\alpha$  - enter the trade. Leave if either TP or SL thresholds are hit.

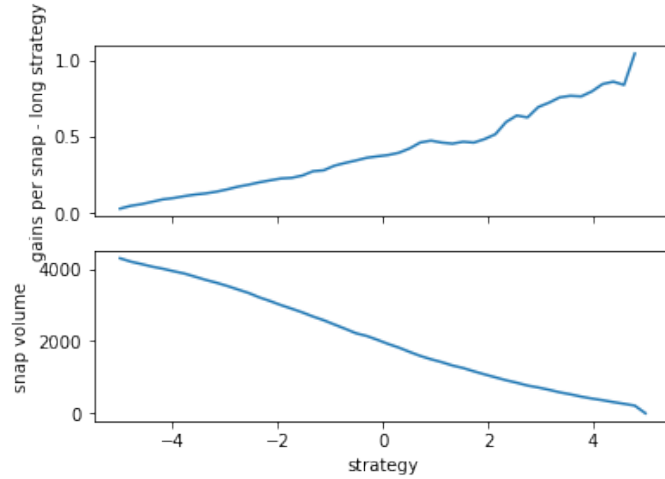


Figure 36: Gains per snapshots vs strategy. Strong correlation between gains and the strategy parameter. The number of snapshots meeting the strategy requirement naturally decreases.

Figure 37 illustrates slight imbalance in the dataset. The strategy to always enter ( $\alpha = -5$ ) makes 101 points, however, it constitutes of only 13.5% of potential 750

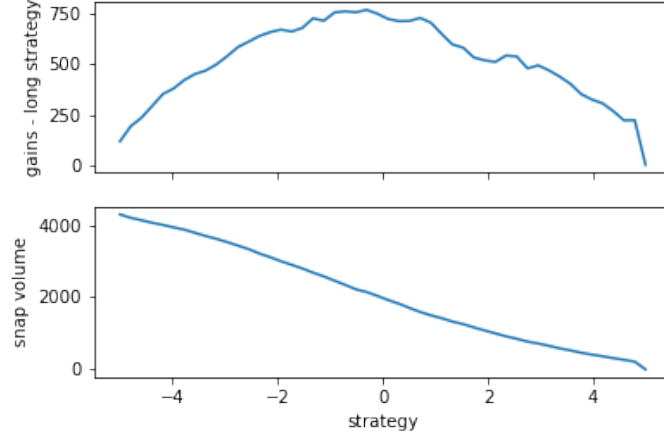


Figure 37: Gains per snapshots vs strategy. Peak of total gains at around  $\alpha = 0$ . The number of snapshots meeting the strategy requirement naturally decreases.

points gains with strategy to buy with  $\alpha = 0$ . One way to explain the imbalance would be to blame the imperfect price normalisation procedure, which leaks some of the positive trend into the price. However, the imbalance effect is not strong enough to explain top gains. There is a clear correlation between  $\alpha$  and gains per snapshot illustrated in Figure 36. One can speculate, that the gains per trade are enough to cover transaction fees, as gain per snapshot can be translated to around 0.42% gain and transaction costs on the Binance exchange vary between 0.02% and 0.07%.

## 9.5 Volatility forecasting

Volatility is a financial quantity associated with risk, most commonly defined as a standard deviation of prices in a given period. In most cases, the higher the volatility, the riskier the underlying asset. Here we define volatility as the standard deviation of paths from 2D snapshots.

To compute volatility directly from the snapshot we generate  $K = 50$  paths from the PDF, calculate their standard deviation in time and average the result. Because of frequent bifurcations (which we can observe in example in Figure 32), the volatility measured this way is naturally inflated which is illustrate in Figure 38.

Figure 38 illustrates vast differences between various volatility indicators that can be read from the forested PDF.

For more clarity, let us inspect only G\_Truth (ground truth) and DMD\_mlp (most likely path). The corresponding histogram is illustrated in Figure 39

Let us now inspect linear correlations between the volatility forecasts. The relations are summaries in Table 1. In addition to already introduced DMD derived values, we added prev\_G\_Truth (previous ground truth) indicator - which is essen-

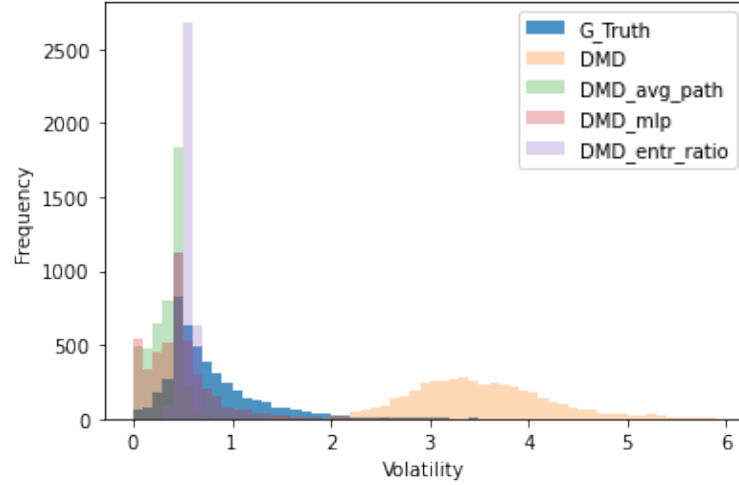


Figure 38: Histogram of various values related to volatility across the test dataset. G\_Truth: true volatility. DMD: volatility estimated directly from the 2D Snapshots and a path generator. DMD\_avg\_path: volatility of an average path in the snapshot. DMD\_mlp: volatility of a most likely path in the snapshot. DMD\_entr\_ratio: entropy of a snapshot, normalised. Most likely and average paths were first introduced in Figure 32.

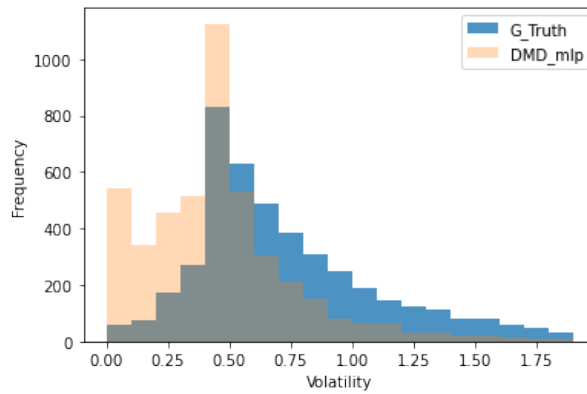


Figure 39: Histogram of values related to volatility across the test dataset. DMD\_mlp: volatility of a most likely path in the snapshot. Reduced version of Figure 38.

tially what RW model would forecast.

It turns out the most effective correlation-wise indicator is DMD\_entr\_ratio (0.3) followed by prev\_G\_Truth (0.28). One could speculate that the hierarchy of methods would change if we would employ a non-linear relationship in place of the linear correlation. This could be true especially for the entropy ratio, given the fact that it does not measure volatility directly. Nevertheless, RW model shows impressive results, given the simplicity of the modeling.



	<b>G_Truth</b>
<b>DMD</b>	0.20
<b>DMD_avg_path</b>	0.10
<b>DMD_mlp</b>	0.23
<b>DMD_entr_ratio</b>	0.30
<b>prev_G_truth</b>	0.28
<b>GP</b>	0.04

Table 1: Linear correlation table between various forecasted PDF-based volatility indicators and volatility of true prices (G\_Truth). Low value for GP is due to constant noise term dominating the volatility.

## 10 Supervised DMD experiment

With the ever-changing environment, data-driven dynamical system modelling can be very challenging due to time-dependencies resulting in non-stationary data distributions. There is a possibility of unknown risks associated with models trained using all accessible data when some of it can no longer be relevant. For example, professionals in finance reduce the scope of training datasets only to relatively recent periods due to potentially critical biases in the forecasting. There is a need to assess which periods of data can still be beneficial, and to understand the rate of its non-stationarity. The proposed idea is to utilise Koopman Theory - a theoretical framework known for strong theoretical guarantees and interpretability, and to combine it with multitask and supervised learning to enable the utilisation of all the available data, and assess forecasting errors due to the non-stationarity of the underlying data distribution.

### 10.1 Seasonal labels

We introduce seasonal labels - time-wise labeling of data regions. We divide existing data not into singular train and test&validation region, but into 8 regions (each around 8 months long) which can exchange their roles to study their differences. If the data stationarity claim/assumption (discussed in Section 5.1) is true, DMD models created using different periods should have similar structures and comparable performances. Let us explore what would happen if we create three models: using region 1, region 2, and combined regions 1&2. The resulting eigenvalues distributions are illustrated in Figure 40.

The general structure of uniform circle and satellites within regions 1 and 2 is preserved, but particular eigenvalues are different. Additionally, the uniform circle gets smaller ones we include more data (combined 1& 2 region datasets). One can speculate that the similar structure in the distributions between regions 1 and 2 come from the fact that the amount of data used to train is the same, and not necessarily point out to some deep connections between underlying datasets. Size

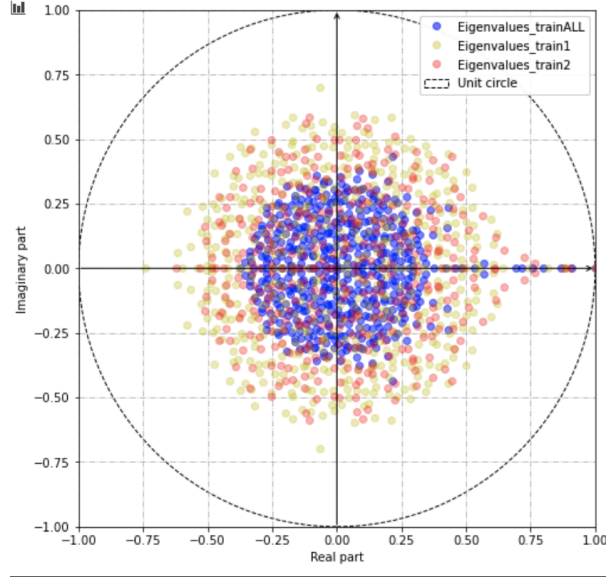


Figure 40: Eigenvalues distribution for three training datasets: 2017-2018 (train1), 2019-2020 (train2) and combined (trainALL). SVD rank selected to cover 92% of the area in the SVD decomposition. The uniform circle gets bigger for smaller training datasets which suggests (similar to 20) a stronger influence of overfitting.

of the uniform circle (as discussed in Section 8.2) suggests overfitting, particularly for singular regions.

We have conducted an additional experiment, where we employ supervised DMD via multitask learning (discussed in Section X) and treating different regions of data as different tasks. The experiment involves comparing regions by measuring the quality of price forecasting applied to neighboring regions. We highlight two pairs of regions: a)  $n - 1 \Rightarrow n$ , b)  $n - 2 \Rightarrow n$ , where  $n \in \{1, \dots, 8\}$  indicates a region. We utilise these region pairs to look for neighbouring region-based correlations. In Figure 41 we illustrate results for DMD, SDMD, and RW. RW model here can serve as a benchmark of which method underperforms a simple forecast. DMD results are straightforward, but SDMD approach needs further explanation. After building the SDMD model using all but one region (the test region), we have to decide which set of eigenvectors (specialised for a particular task/ region) we want to use. The one that we end up using is the one highlighted by previous or previous previous label.

Conclusion we draw from Figure 41 is that, that it illustrates some neighbouring region-based correlations. Previous models outperform the previous previous ones, which is an indication of short term dependencies. SDMD method clearly beats DMD in this experiment, however it is important to point out that although the eigenvectors in SDMD are specialised to a singular region, the eigenvalues contain information gathered across 7 regions, as opposed to DMD which does not utilise any additional information besides a singular region.

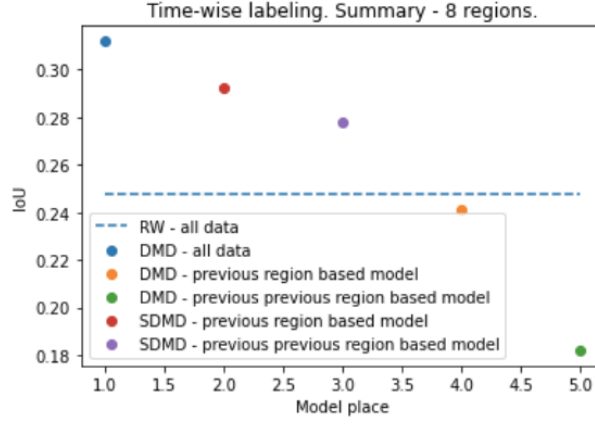


Figure 41: IoU performance for various prediction methods. RW model serves as a benchmark. The dataset is divided into 8 regions. Previous region based model results is an average result of models train using only one region which is directly before the region which the model is forecasting. Similarly, previous previous region based models train using a region two periods before the test one. Previous models heavily outperform previous previous models which reveals region-based differences thus non-stationarity of data distributions.

However, none of the models were able to outperform DMD utilising 7 regions, which suggests that more than simple short-term dependencies and/or overfitting via specialisation to a relatively small data-points-wise region.

## 11 Summary and conclusions

### 11.1 Summary

Our main results consists of: a) formulation of a useful normalisation method to fulfil DMD algorithm requirements of a locally linear map and reducing the scope of non-stationarity of the data distributions between regions, b) formulation of the mismatch index, which enables a search of many hyper-parameters without favouring a particular modeling method, and suggests that price forecasting should be theoretically possible, c) sine wave with noise data analysis which builds intuition around potential interpretations of DMD eigenvectors and eigenvalues, d) financial data price forecasting highlighting strengths of each of the evaluated methods, e) financial data analysis related to data non-stationarity, where we utilise Koopman theory and multitask learning to better understand differences between training datasets

## 11.2 Conclusions

The main goal of developing an appropriate implementation of DMD, SDMD, and drawing broad inferences from financial data models was fulfilled.

Alongside regression and classification tasks, we explored interpretability of the Koopman theory derived models. We found that Koopman based DMD and benchmark methods GP, RW have its own forecasting strengths, and one can speculate that in a general case, the DMD model would be the most appropriate. Additionally, the DMD model is able to accurately classify potentially beneficial investments, even with hypothetical transaction fees. Lastly, we explored Koopman theory combined with multitask learning to discuss stationarity of the distribution of the studied data. We discovered short-term time dependencies between neighboring data regions.

# Bibliography

- [1] J. Nathan Kutz et al. *Dynamic Mode Decomposition*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2016. DOI: [10.1137/1.9781611974508](https://doi.org/10.1137/1.9781611974508). eprint: <https://epubs.siam.org/doi/pdf/10.1137/1.9781611974508>.
- [2] Jonathan H. Tu et al. “On dynamic mode decomposition: Theory and applications”. In: *Journal of Computational Dynamics* 1.2 (2014), pp. 391–421. DOI: [10.3934/jcd.2014.1.391](https://doi.org/10.3934/jcd.2014.1.391).
- [3] Jonathan H. Tu et al. “On dynamic mode decomposition: Theory and applications”. In: *Journal of Computational Dynamics* 1.2 (2014), pp. 391–421. DOI: [10.3934/jcd.2014.1.391](https://doi.org/10.3934/jcd.2014.1.391).
- [4] PETER J. SCHMID. “Dynamic mode decomposition of numerical and experimental data”. In: *Journal of Fluid Mechanics* 656 (2010), pp. 5–28. DOI: [10.1017/S0022112010001217](https://doi.org/10.1017/S0022112010001217).
- [5] J. Nathan Kutz Dirk M. Luchtenburg Steven L. Brunton. “On dynamic mode decomposition: Theory and applications”. In: *Journal of Computational Dynamics* 1.2 (2014), pp. 391–421.
- [6] Naftali Cohen et al. *Visual Time Series Forecasting: An Image-driven Approach*. 2021. DOI: [10.48550/ARXIV.2107.01273](https://doi.org/10.48550/ARXIV.2107.01273).
- [7] Marcos Lopez de Prado. *Advances in Financial Machine Learning*. Wiley Publishing, 2018.
- [8] Marcos López de Prado. *Interpretable Machine Learning: Shapley Values (Seminar Slides)*. pt. Available at SSRN: June 27, 2020. DOI: [10.2139/ssrn.3637020](https://doi.org/10.2139/ssrn.3637020). URL: <http://dx.doi.org/10.2139/ssrn.3637020>.
- [9] Mihailo R. Jovanović, Peter J. Schmid, and Joseph W. Nichols. “Sparsity-promoting dynamic mode decomposition”. In: *Physics of Fluids* 26.2 (2014), p. 024103. DOI: [10.1063/1.4863670](https://doi.org/10.1063/1.4863670).
- [10] In: *Introduction to Statistical Machine Learning*. Ed. by Masashi Sugiyama. Boston, 2015. ISBN: 978-0-12-802121-7. DOI: <https://doi.org/10.1016/B978-0-12-802121-7.00052-2>.
- [11] Keisuke Fujii and Yoshinobu Kawahara. “Supervised dynamic mode decomposition via multitask learning”. In: *Pattern Recognition Letters* 122 (2019), pp. 7–13. DOI: <https://doi.org/10.1016/j.patrec.2019.02.010>.

- [12] Andreas Mardt et al. “VAMPnets for deep learning of molecular kinetics”. In: *Nature Communications* 9.1 (Jan. 2018), p. 5. DOI: [10.1038/s41467-017-02388-1](https://doi.org/10.1038/s41467-017-02388-1).
- [13] Christoph Wehmeyer and Frank Noé. “Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics”. In: *The Journal of Chemical Physics* 148.24 (2018), p. 241703. DOI: [10.1063/1.5011399](https://doi.org/10.1063/1.5011399). eprint: <https://doi.org/10.1063/1.5011399>.
- [14] Samuel E. Otto and Clarence W. Rowley. “Linearly Recurrent Autoencoder Networks for Learning Dynamics”. In: *SIAM Journal on Applied Dynamical Systems* 18.1 (2019), pp. 558–593. DOI: [10.1137/18M1177846](https://doi.org/10.1137/18M1177846). eprint: <https://doi.org/10.1137/18M1177846>.
- [15] Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. “Deep learning for universal linear embeddings of nonlinear dynamics”. In: *Nature Communications* 9.1 (Nov. 2018), p. 4950. DOI: [10.1038/s41467-018-07210-0](https://doi.org/10.1038/s41467-018-07210-0).
- [16] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time series analysis: forecasting and control*. 4th ed. Wiley series in probability and statistics. OCLC: ocn176895531. Hoboken, N.J: John Wiley, 2008. ISBN: 9780470272848.
- [17] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. en. The MIT Press, 2005. ISBN: 9780262256834. DOI: [10.7551/mitpress/3206.001.0001](https://direct.mit.edu/books/book/2320/gaussian-processes-for-machine-learning). URL: <https://direct.mit.edu/books/book/2320/gaussian-processes-for-machine-learning> (visited on 07/17/2021).
- [18] David Duvenaud. “Automatic model construction with Gaussian processes”. PhD thesis. Nov. 2014. URL: <https://www.cs.toronto.edu/~duvenaud/thesis.pdf>.
- [19] Eugene F. Fama. “Efficient Capital Markets: A Review of Theory and Empirical Work”. In: *The Journal of Finance* 25.2 (1970), p. 383. DOI: [10.2307/2325486](https://doi.org/10.2307/2325486).
- [20] Burton G. Malkiel. “The Efficient Market Hypothesis and Its Critics”. In: *Journal of Economic Perspectives* 17.1 (Mar. 2003), pp. 59–82. DOI: [10.1257/089533003321164958](https://doi.org/10.1257/089533003321164958).
- [21] Bloomberg. *How an Exclusive Hedge Fund Turbocharged Its Retirement Plan*. 2015. URL: <https://www.bloomberg.com/news/articles/2015-06-16/how-an-exclusive-hedge-fund-turbocharged-retirement-plan>.
- [22] Jim Simons. *The mathematician who cracked Wall Street*. 2015. URL: [https://www.ted.com/talks/jim\\_simons\\_the\\_mathematician\\_who\\_cracked\\_wall\\_street](https://www.ted.com/talks/jim_simons_the_mathematician_who_cracked_wall_street).
- [23] Dirk Willenbockel. *The Price Normalisation Problem in General Equilibrium Models with Oligopoly Power: An Attempt at Perspective*. DOI: [10.2139/ssrn.552764](https://doi.org/10.2139/ssrn.552764).

- [24] Stefano Martiniani et al. *Correlation Lengths in the Language of Computable Information*. DOI: [10.1103/physrevlett.125.170601](https://doi.org/10.1103/physrevlett.125.170601).
- [25] Michal Balcerak and Thomas Schmelzer. *Constructing trading strategy ensembles by classifying market states*. DOI: [10.48550/ARXIV.2012.03078](https://doi.org/10.48550/ARXIV.2012.03078).
- [26] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. “Practical Bayesian Optimization of Machine Learning Algorithms”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 2951–2959. URL: <http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf>.
- [27] Lisha Li et al. “Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization”. In: *J. Mach. Learn. Res.* 18 (Jan. 2017), pp. 6765–6816. ISSN: 1532-4435.